Plane-based 3D Mapping for Structured Indoor Environment

POLITECNICO DI TORINO

SCUOLA DI DOTTORATO
PhD in Mechatronics – XXV cycle

PhD dissertation

Plane-based 3D Mapping for Structured Indoor Environment

Zehui Yuan

Supervisor
Prof. Basilio Bona

Doctoral Course Coordinator
Prof. Giancarlo Genta

February 2013

Acknowledgement

I wish to express my gratitude to the following people: First of all, I would like to thank Prof. Basilio Bona for offering me the possibility to study in his great team in a friendly atmosphere. The last three years at his lab were a great time and I especially want to express my appreciation for giving me the support and guidance.

Secondly, I want to address special thanks to my colleagues, Dr. Luca Carlone, Stefano Rosa and Vito Macchia for their advices and support. During the last three years, their help was available at all times. Without their help, this work would not be possible. I also wish to thank other colleagues in the lab who already left the group for continuing their successful career.

Last but not least, I would like to thank all my friends and my family, especially my parents for their love and constant support. They were always there for me, even in the dark times of frustrations and failure, and hardships of my studies.

Contents

List of Figures

List of Tables

Abstract

Three-dimensional (3D) mapping deals with the problem of building a map of the unknown environments explored by a mobile robot. In contrast to 2D maps, 3D maps contain richer information of the visited places. Besides enabling robot navigation in 3D, a 3D map of the robot surroundings could be of great importance for higher-level robotic tasks, like scene interpretation and object interaction or manipulation, as well as for visualization purposes in general, which are required in surveillance, urban search and rescue, surveying, and others.

Hence, the goal of this thesis is to develop a system which is capable of reconstructing the surrounding environment of a mobile robot as a three-dimensional map.

Microsoft Kinect camera [67] is a novel sensing sensor that captures dense depth images along with RGB images at high frame rate. Recently, it has dominated the stage of 3D robotic sensing, as it is low-cost, low-power. For this work, it is used as the exteroceptive sensor and obtains 3D point clouds of the surrounding environment. Meanwhile, the wheel odometry of the robot is used to initialize the search for correspondences between different observations.

As a single 3D point cloud generated by the Microsoft Kinect sensor is composed of many tens of thousands of data points, it is necessary to compress the raw data to process them efficiently. The method chosen in this work is to use a feature-based representation which simplifies the 3D mapping procedure.

The chosen features are planar surfaces and orthogonal corners, which is based on the fact that indoor environments are designed such that walls, ground floors, pillars, and other major parts of the building structures can be modeled as planar surface patches, which are parallel or perpendicular to each other. While orthogonal corners are presented as higher features which are more distinguishable in indoor environment.

In this thesis, the main idea is to obtain spatial constraints between pairwise frames by building correspondences between the extracted vertical plane features and corner features. A plane matching algorithm is presented that maximizes the similarity metric between a pair of planes within a search space to determine correspondences between planes. The corner matching result is based on the plane matching results. The estimated spatial constraints form the edges of a pose graph, referred to as graph-based SLAM front-end.

In order to build a map, however, a robot must be able to recognize places that it has previously visited. Limitations in sensor processing problem, coupled with environmental ambiguity, make this difficult. In this thesis, we describe a loop closure detection algorithm by compressing point clouds into viewpoint feature histograms, inspired by their strong recognition ability. The estimated roto-translation between detected loop frames is added to the graph representing this newly discovered constraint.

Due to the estimation errors, the estimated edges form a non-globally consistent trajectory. With the aid of a linear pose graph optimizing algorithm, the most likely configuration of the robot poses can be estimated given the edges of the graph, referred to as SLAM back-end. Finally, the 3D map is retrieved by attaching each acquired point cloud to the corresponding pose estimate. The approach is validated through different experiments with a mobile robot in an indoor environment.

Keywords:

Mobile robot Mapping, 3D point cloud modeling, Structured Environment Mapping, Pose-graph SLAM, Plane Extraction, Plane matching, Microsoft Kinect

Chapter 1

Introduction

Today robotic systems are widely used in the industry, in particular for tasks such as welding, painting and packaging. All of these robot systems are in the form of manipulators that carry out repetitive motions. For large scale transformations such robotic systems are not particularly practical. Recent advances in mobile robotics have allowed widespread use of the robot in several applications. For example, a large number of mobile robots have been built for developing tasks in search, rescue, and exploration to perform in dangerous area for human being. And another potentially interesting area where mobile robots are used is the service sector. In this case, the robots perform tasks in indoor environment autonomously that relieve the human being, such as health care, cleaning and entertainment. The fact that robots are rapidly evolving from factory work-horses to robot companions poses a great challenge for the future of robots: they must be capable of coping with complex tasks and working in dynamic environments.

Intelligent behavior and interaction with the outside environment for a mobile robot requires understanding the geometry and structure of the environment, i.e., a representation of its surroundings that adequately resembles the spatial properties of the environment. Such representation of the surrounding scenario is called a *map*. It appears to be the minimum amount of spatial abstraction required for an autonomous mobile robot. Simultaneous Localization and Mapping (SLAM), originally introduces an environment while at the same time localizing the mobile robot relative to the map under construction, given a sequence of measurements gathered by its proprioceptive and exteroceptive sensors. From a mathematic point of view, SLAM is difficult to solve since the mapping and robot poses estimating procedures are generally dependent and can not be obtained separately.

Today in the mobile robotic research community, it is well agreed that SLAM is an important requirement for intelligent mobile robots. It has attracted a conspicuous attention from the robotics community for its vast application domain, and robot

mapping has predominantly been investigated in 2D. Now with the availability of new kinds of 3D sensors, e.g., laser sensors, stereo camera sensors and Microsoft Kinect, 3D map is becoming popular and robots are increasingly operating in 3D environment. Meanwhile, in contrast to a 2D grid map, 3D map contains a rich description of the environment and is able to provide more information for the robot. There is hence an increasing amount of research on 3D mapping, especially on 3D SLAM.

Typical 3D sensors can be categorized as: (1) laser range finders, is usually mounted on a rotating platforms [87] having a large scanning time of around a minute. Laser range finder can attach the environment in a large field-of-view (FOV) and obtain the environment information precisely. (2) time-of-flight (TOF) sensors like the Swiss-range [16] and PMD [74]. Compared with laser range finder, it has a much restricted FOV, but it is being able to provide several scans per second. (3) stereo cameras [5] [54]. They are inexpensive, and provide high information bandwidth of the environment.

With the introduction of the Microsoft Kinect camera, a new sensor has appeared on the market that provides both RGB images along with perpixel depth information. It allows the capture of reasonably accurate mid-resolution depth and appearance information at high data rates. Meanwhile, it is low-cost and low power. Thus it is attractive for the research outside specialized computer vision groups and has dominated the stage of 3D robotic sensing. In this regard, in this thesis, our efforts are aimed to construct 3D maps of the structured indoor environments with a robot only equipped with a 3D Kinect camera and wheel encoders.

1.1 3D Mapping

3D mapping is concerned with the problem of building a map of an unknown environment explored by a mobile robot. Most 3D mapping systems contain three main components: (1) the spatial alignment of consecutive data frames; (2) the detection of loop closures; (3) the globally consistent alignment of the complete data sequence. The first two components obtain the pose changes between consecutive data frames and arbitrary frames, which are modeled as edges (or constraints) between the related nodes in a graph (the pose graph). Each node in a graph represents a robot pose (or frame). This procedure is usually refereed to as as pose graph construction (graph-based SLAM front-end). While the third one is the so called pose graph optimization (graph-based SLAM back-end) to determine the most likely configuration of the poses given the edges of the graph, hence obtaining an accurate estimate of the poses assumed by the robot. Then the 3D map is created by attaching

data sets gathered by its exteroceptive sensors to their corresponding poses into a global coordinate frame.

In principle, a robot equipped with a 3D camera and wheel encoders would be able to create a 3D dense map of the environment by attaching the point clouds to the corresponding poses estimated from wheel odometry. However, this naive strategy quickly becomes inaccurate, since wheel odometry suffer from error accumulation. With typical odometry errors the pose estimate will be totally wrong after as little as 10 m of travel [7]. Scan-matching approaches were used successfully to address the robot poses tracking problem. The idea is to align consecutive scans taken by the external sensors from the robot at different locations and thereby estimate the relative pose offset of the robot between two successive range sensor samples. The most commonly used scan-matching algorithm is the point-to-point (P-P) iterative closest point (ICP) and its variants. A recent dissertation [92] notes that "up to now, all approaches successfully applied to 3D SLAM are based on the ICP algorithm.

Alignment between successive frames is a good method for tracking the robot position over moderate distances. However, errors in alignment between a particular pair of frames, and noise and other kinds of errors, cause the estimation of camera position to drift over time, leading to an consistent and inaccuracy map. This is most noticeable when the camera follows a long path. The cumulative error in frame alignment results in a map that has two representations of the same region in different locations. This is known as the loop closure problem, which is critical for the poses global optimization, since the loop closing allows to reduce the accumulated error.

In this thesis, we will follow the overall structure of recent 3D mapping techniques, but we introduce a new approach that is different from the overall approaches found in the literature, as presented in the following section, while it differs from existing approaches.

1.2 Goal and Contributions

As mentioned, the goal of this thesis is to build the 3D maps of structured 3D indoor environments by developing a plane-based SLAM approach and validate it experimentally. And a Microsoft Kinect is used to generate dense 3D models of indoor environments. It is based on the fact that a major part of indoor environments can be represented by sets of planar patches and orthogonal corners. Targeting the application of mobile robots in 3D scenarios, special effort has been put in development of proper 3D representations of the surrounding environment around the robot due to the availability of 3D sensors. In contrast to occupancy grid maps, 3D maps include

more detailed information. For mobile robots, the knowledge of a map containing a rich description of the environment is very helpful for navigation, surveying tasks and manipulation, especially when the robot has to operate in 3D scenarios. Instead of more accurate laser sensors, here the Microsoft Kinect is used as the exteroceptive sensor for 3D perception since it is low-cost, low-power and is able to acquire color images and depth maps at full frame rate.

Major contributions of this thesis are:

1. A plane matching algorithm is presented to create constraints among the poses assumed by the robot, i.e., SLAM *front-end*. The planar segments is used as the basic feature approximating underlying point clouds generated by Microsoft Kinect. Using the proposed matching algorithm, the geometric constraints between extracted planes sets from pairwise frames are retrieved.

2. A corner matching algorithm is presented which is based on the plane matching result. Beside the plane segments, also orthogonal corners are dominant features in indoor environments. Moreover, they are more robust and distinguishable since they can lock all degrees of freedom in space. Based on this, all the corners in a frame are detected according to the relationship between the extracted planes. Combined with the plane matching results, the correspondences between corners in consecutive frames are determined.

   Based on the built corresponding relationship between planes and corners, the relative roto-translation between the registered frames are estimated, which form the edges of a pose graph.

3. A new loop closing detection technique is introduced in order to diminish the accumulated error. This technique is built by combining a new 3D geometry descriptor, View Point Histogram (VFH) presented in [78] and appearance based features. The main idea here comes from the strong recognition ability of VFH. In order to make the loop detection more robust and reliable, appearance based feature, color histogram is used to compare the similarity of the detected frames. When the pairwise frames are determined as a same area, their relative roto-translation information is estimated and added to the built pose graph. The pose graph is then be feed to a pose graph optimizer to obtain a globally and consistent trajectory of robot.

1.3 Thesis Outline

In the next Chapter, after presenting a brief background on the graph-based SLAM foundation, the state-of-the-art is reviewed to give a picture of existing solution to SLAM front-end and back-end respectively. Moreover, two kinds of popular algorithms for SLAM front-end, iterative closest point (ICP) and sample consensus initial alignment (SAC-IA) are briefly introduced, which will be used to compare with the technique presented in this thesis.

In the ensuing chapters, the plane-based SLAM is presented which is based on constructing the matching relationship between planar segments sets extracted from 3D point clouds obtained from a Microsoft Kinect camera. Obviously, these planes have to be extracted from the raw data, which will be dealt in Chapter 3. Since the raw point clouds suffer from different noise and error sources. Therefore pre-possessing procedure is performed firstly. Then the RANSAC plane model is selected to extract planes which meeting our pre-determined criterion. Moreover, in order to make the extracted planar segments accurate, plane refinement procedures are followed. Afterward, orthogonal corners are detected based on the planes extraction results.

Chapters 4 focus on the pose-graph SLAM front-end part. The plane matching and corner matching algorithm is presented thoroughly, which consists in finding correspondences between planar surface segments and orthogonal corners in the two scans to be matched. After the correspondences have been decided on, the relative rotation and translation that aligns the corresponding set of planes and corners will be estimated. This gives the pose changes of the robot between the scans, which form a pose graph.

Then in Chapter 5, loop closure detection is addressed using 3D point clouds and the pose graph optimization, refereed to as SLAM back-end, are discussed. It starts with a brief and general review of existing approaches of loop closure detection algorithms. Then a loop closure detection algorithm based on viewpoint feature histogram (VFH), is explained thoroughly. To obtain a globally consistent trajectory, targeting to this work, a linear pose-graph optimization, proposed in [12] [11] is selected to optimize the built pose-graph.

Finally Chapter 6 includes experimental results used to evaluate the developed plane-based 3D mapping algorithm. The experiments are carried out in three different scenarios inside Dipartimento di Automatica e Informatica at Politecnico Di Torino. Meanwhile, to estimate the performance of the plane matching approach, pairwise registration tests between several sets of successfully paired consecutive frames are preformed firstly. And the registration results are compared to two pop-

ular registration algorithms ICP and SACIA.

The thesis ends with a short chapter on conclusions of this thesis, and suggestions for future work and development.

Chapter 2

State-of-the-art

2.1 Introduction

3D mapping is the process of building a map of the environment where the robots operate. It is retrieved by attaching the sensor measurements into their corresponding poses of the mobile robot. Thus a known robot pose is required for the 3D mapping. In outdoor spaces, this is possible since Global Positioning System (GPS) is available that provides an absolute position around the globe with centimeter range precision in ideal conditions. However, in indoor environments, GPS signal is disturbed or not available. In these cases, reliable mapping using GPS can not be carried out. The acquisition of maps of indoor environment has been a major research focus in the robotics community over the last decades.

Generally, in indoor environments, robot poses are provided by internal sensors e.g., wheel encoders or IMUs. However, these sensors accumulate errors as the mobile robot explores, therefore can only be used reliably over short distances; for example, the pose estimated by wheel odometry will be totally wrong after as little as 10 m of travel. Learning maps under pose uncertainty is often referred to as the simultaneous localization and mapping (SLAM) problem. In the literature, a large number of solutions to this problem is available.

The first method, building the basis of SLAM algorithms, was presented in [83], establishing a statistical basis for describing geometric uncertainty and relationships between features or landmarks. Based on it, several other working solutions to the SLAM problem were described: presently the extended Kalman filter [14] [62] based SLAM is the most frequently used approach. Other popular techniques include information filters [23] [90] and particle filters [35] [38]. The SLAM is modeled as an online state estimation where the system state consists of the current robot position and the map. The map and robot poses are augmented and updated

by including newly arrived sensor measurements.

An intuitive way to address the SLAM problem is via the so-called graph-based formulation, first proposed by Lu and Milios in 1997 [64]. Solving a graph-based SLAM problem involves to construct a graph whose nodes represent robot poses or landmarks and in which an edge between two nodes represent inter-nodal measurements that constraints the connected poses. For instance, odometric measurements are modeled as edges (or constraints) connecting consecutive nodes, while loop closing edges connect arbitrary nodes pairs and model place revisiting episodes. Once such a graph is constructed, the critical problem is to find the configurations of poses that maximize the likelihood of the inter-nodal measurements. This involves solving a large error minimization problem. Thus, in graph-based SLAM the problem is decoupled in two tasks: constructing the graph from the sensor measurement (*graph construction*), determining the most likely configuration of the poses given the edges of the graph (*graph optimization*). The graph construction is usually called front-end and it is heavily sensor dependent, while the second part is called back-end and relies on an abstract representation of the data which is sensor agnostic [33].

In this Chapter, firstly a brief background on the graph-based SLAM foundation is presented. Then state-of-the-art is reviewed to give a picture of existing solution to SLAM front-end and back-end respectively. Moreover, two kinds of popular algorithms for SLAM front-end, iterative closest point (ICP) and sample consensus initial alignment (SAC-IA) are briefly introduced. In Chapter 6 they will be considered as standard algorithms to compare with the approach presented in this thesis.

2.2 Graph-based SLAM

2.2.1 SLAM problem

Assuming a robot is moving in an unknown environment, simultaneous localization and mapping (SLAM) is concerned about building a map of the surrounding environment and estimating the robot trajectory at the same time. Due to the inherent noise of the sensor measurements, usually the SLAM problem is described by means of probabilistic tools. The dominant scheme used in SLAM is the Bayes filter. The Bayes filter extends Bayes rule to temporal estimation problems. It is a recursive estimator for computing a sequence of posterior probability distributions over quantities that can not be observed directly, such as a map.

For the reason of explaining, the robot's trajectory is described by the sequence of random variables $\mathbf{x}_{1:n} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, where $\mathbf{x}_i$ is the robot's pose at time i.

When the robot is moving, it acquires a sequence of odometry measurements $\mathbf{u}_{1:n} = \{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n\}$ and perceptions of the environment $\mathbf{z}_{1:n} = \{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n\}$. Solving the full problem consists of estimating the posterior probability of the robot's trajectory $\mathbf{x}_{1:n}$ and the map $\mathbf{m}$ of the environment given all the measurements plus an initial position $\mathbf{x}_0$:

$$p(\mathbf{x}_{1:n}, \mathbf{m} | \mathbf{z}_{1:n}, \mathbf{u}_{1:n}, \mathbf{x}_0) \tag{2.1}$$

The initial position $\mathbf{x}_0$ that defines the position of the map $\mathbf{m}$ can be chosen arbitrarily. By convention, it is usually set to be the origin of the global reference frame, i.e., $\mathbf{x}_0 = [0 \;\; 0 \;\; 0]^\top$.

To solve the SLAM problem, the robot needs to be endowed with models that describe the effect of the control input and the observations, that is, a state transition model and an observation model, respectively.

The observation model describes the probability of making an observation $\mathbf{z}_i$ when the vehicle location and landmark locations are known. It is assumed that, once the vehicle location and map are defined, observations are conditionally independent given the map and current vehicle state. Generally, the observation model is described in the form

$$p(\mathbf{z}_i | \mathbf{x}_i, \mathbf{m}). \tag{2.2}$$

The motion model for the vehicle is described in terms of a probability distribution on state transitions in the form

$$p(\mathbf{x}_i | \mathbf{x}_{i-1}, \mathbf{u}_i) \tag{2.3}$$

That is, the state transition is assumed to be a Markov process in which the next state $\mathbf{x}_i$ depends only on the immediately proceeding state $\mathbf{x}_{i-1}$ and the applied control $\mathbf{u}_i$, and is independent of both the observations and the map.

2.2.2 Graph-based SLAM

A large variety of SLAM approaches are available in the robotics community. For instance, Kalman Filter, Particle Filters and Graph-based SLAM. Recently graph-based SLAM has attracted a conspicuous attention from the robotics community, since it highlights a spatial structure and is well suited to describe filtering processes of SLAM.

In graph-based SLAM, the poses of the robot are modeled by nodes in a graph and labeled with their positions in the environment [55][64]. Spatial constraints between poses that result from observations $\mathbf{z}_i$ or from odometry measurements $\mathbf{u}_i$ are encoded in the edges between the nodes. In more details, a graph-based SLAM algorithm constructs a graph out of the raw sensor measurements. Figure 2.1

illustrates a presentation of pose-graph SLAM process. Every node in the graph stands for a robot pose at which a sensor measurement was acquired, and each edge between two nodes encodes the spatial information arising from the alignment of the connected measurements and can be regarded as a spatial constraint relating these two poses. An edge between two nodes consists in a probability distribution over the relative locations of the two poses, conditioned to their mutual measurements.

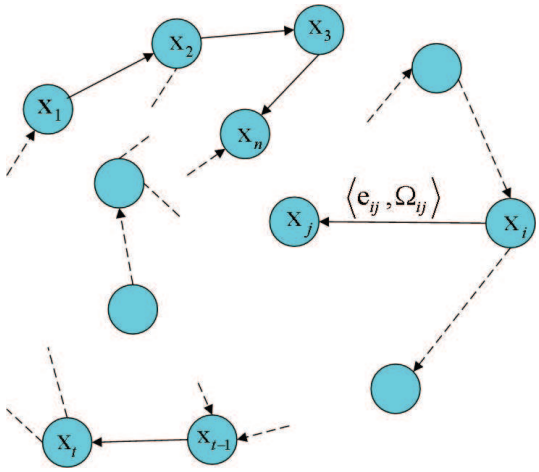


Figure 2.1: Every node stands for a robot pose. Adjacent nodes are connected by edges that represent inter-nodal measurements. The edges are divided into two classes: (1) edges between consecutive poses, obtained by odometry or scan-matching; (2) edges between non-consecutive poses, arising from multiple observations of the same part of the environment.

Generally, the observation model $p(\mathbf{z}_i \mid \mathbf{x}_i, \mathbf{m})$ is a multi-modal distribution, which means that a single observation $\mathbf{z}_i$ might result in multiple potential edges connecting different poses in the graph and the graph connectivity needs itself to be described as a probability distribution. To avoid the computation complexity introduced by multi-modality, usually, the estimate is restricted to the most likely topology, i.e., determining the most likely constraint resulting from an observation. This decision depends on the probability distribution over the robot poses. This problem is known as data association and is usually addressed by the SLAM front-end.

As mentioned before, a graph-based SLAM is typically concerned with two problems. The first one is the SLAM front-end, discussed above. It directly deals with the sensor data and interprets the sensor data to extract the spatial constraints. While the second problem is to correct the poses of the robot to obtain a globally consistent map or trajectory given the estimated constraints. This part of the approach is often referred to as the optimizer or the SLAM back-end.

Let us call $\mathbf{x}_{1:n} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, where $\mathbf{x}_i$ describes the pose of node i. Let $\hat{\mathbf{z}}_{ij}$ be the true (unknown) relative transformation between node i and node j. The log likelihood l_{ij} of a measurement $\mathbf{z}_{ij}$ is therefore

$$\mathrm{l}_{ij} \propto [\mathbf{z}_{ij} - \hat{\mathbf{z}}_{ij}]^\top \Omega_{ij} [\mathbf{z}_{ij} - \hat{\mathbf{z}}_{ij}] \tag{2.4}$$

where $\mathbf{z}_{ij}$ and Ω_{ij} represent respectively the mean and the information matrix of a constraint relating the parameters $\mathbf{x}_j$ and $\mathbf{x}_i$.

Define a vector error function $\mathbf{e}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{z}_{ij})$ that measures the difference between the expected observation $\hat{\mathbf{z}}_{ij}$ and the real observation $\mathbf{z}_{ij}$ gathered by the robot, i.e.,

$$\mathbf{e}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{z}_{ij}) = \mathbf{z}_{ij} - \hat{\mathbf{z}}_{ij} \tag{2.5}$$

For simplicity of notation, we will encode the measurement in the indices of the error function:

$$\mathbf{e}(\mathbf{x}_i, \mathbf{x}_j, \mathbf{z}_{ij}) = \mathbf{e}_{ij}(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{e}_{ij} \tag{2.6}$$

Figure 2.2 presents the error functions and the quantities that play a role in defining an edge of the graph.

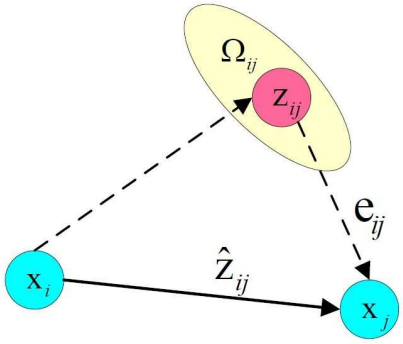


Figure 2.2: The components of an edge connecting the node $\mathbf{x}_i$ and node $\mathbf{x}_j$. $\mathbf{z}_{ij}$ is the real measurement presented in the local reference frame of $\mathbf{x}_i$, while $\hat{\mathbf{z}}_{ij}$ is the expected measurement that makes the data samples $\mathbf{x}_i$ and $\mathbf{x}_j$ perfectly overlapped. The error $\mathbf{e}_{ij}$ depends on the displacement between the expected and the real measurement. An edge is fully characterized by its error function $\mathbf{e}_{ij}$ together with the information matrix of the measurement that accounts for its uncertainty.

The goal of a maximum likelihood approach is to find the configuration $\mathbf{x}^*$ of the nodes that maximizes the log likelihood $\mathbf{F}(\mathbf{x})$ of all the observations

$$\mathbf{F}(\mathbf{x}) = \sum_{(i,j)\in\varepsilon} \underbrace{\mathbf{e}_{ij}^\top \Omega_{ij} \mathbf{e}_{ij}}_{\mathbf{F}_{ij}} \tag{2.7}$$

where ε is the graph edge set, containing the unordered node pairs (i, j) such that a relative pose measurement exists between i and j. Thus, SLAM seeks to solve the following problem:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \mathbf{F}(\mathbf{x}) \tag{2.8}$$

2.3 State-of-the-art of SLAM Front-end

Scan-matching approaches were used successfully to estimate the relative poses between consecutive scans by aligning the data samples taken by the external sensors from the mobile robot at different problem.

The most commonly used scan-matching algorithm is the point-to-point (P-P) iterative closest point (ICP), presented in [4], which works directly with the points and hence does not assume any specific structure in the environment. Since the introduction of basic ICP, a large number of variants have been developed. For instance, point-to-plane (P-L) version of ICP [94][15]. Instead of point-to-point distance cost function, in [94][15], the distance between a point and a planar approximation of the surface at the corresponding point distance measures was used. If there is a good initial position estimate and a relatively low noise, ICP method with the point-to-plane metric has a fast convergence rate. However, when those conditions can not be guaranteed, the point-to-plane ICP is prone to fail [29].

In order to reduce the search space of the ICP algorithm, the iterative closest compatible (ICCP) algorithm was proposed in [30]. In the ICCP algorithm, the distance minimization is performed only between the pairs of points considered compatible on the basis of their viewpoint invariant attributes (curvature, color, normal vector, etc.). Another method, called ICP using invariant feature (ICPIF), was presented in [82]; it chooses the nearest neighbor correspondences by a distance metric that represents a scaled sum of the potential and feature distances. These two algorithms falls into the category of feature-based ICP. Invariant features can be points, lines and other shapes and objects etc. Compared with traditional ICP, feature-based ICP converges to the goal value in fewer iterations, thus the computation time is reduced.

Feature-based methods are commonly used for registration in visual SLAM . Most approaches rely on the extraction and matching of sparse 2D visual features from the camera images. Visual feature points have the advantage of being more informative which simplifies data association. Scale invariant feature transform (SIFT), introduced in [63] has been shown as one of the best-known keypoint descriptor. Its invariance to image translation, scaling and partial invariance to rotation, illumination changes and affine or 3D projection, makes them suitable for

mobile robot localization and map building. In stereo systems, these landmarks are localized and robot ego-motion is estimated by least-squares minimization of the matched landmarks. Feature viewpoint variation and occlusion are taken into account by maintaining a view direction for each landmark. Experiments showed that these visual landmarks are robustly matched, robot pose is estimated and a consistent 3D map is built.

A drawback of SIFT is the high dimensionality of the descriptor, resulting in an increase of computation time. A recent approach, named the Speeded-Up Robust Features (SURF) [2], reduces the computation time and increases the robustness. These are achieved by using a very basic approximation, i.e., assuming second order Gaussian derivatives with box filters, and describing a distribution of Haar-wavelet responses within the interest point neighborhood. The experiments show that SURF can be computed more efficiently and yields a lower dimensional feature descriptor, so that the matching procedure is faster. In [69], Murilo *et al.* proposed the use of SURF to improve the appearance-based localization and mapping methods that perform image retrieval in large data sets. In their experiments they compared SURF algorithm with SIFT using omnidirectional images. The experiments showed that the use of SURF offers the best compromise between efficiency and accuracy, and performs always the best or being much faster in case of similar accuracy.

The SIFT and SURF methods described above, mainly focus on the 2D points. Recently others 3D feature descriptors have been presented, e.g., Normal Aligned Radial Feature (NARF) descriptor [86], and Fast Point Feature Histogram (FPFH) [76].

In indoor SLAM, 2D lines and 3D planar surfaces are popular used as matching features, since the common indoor environment is made up by many planar surfaces. Compared to points, 2D lines and 3D surfaces are more distinguishable. Plane features can be automatically extracted from point clouds with surface growing methods or RANSAC plane model, while line features are usually extracted from the intersection of plane features.

Horn *et al.* [45] presented early work on using 3D data for robot navigation, extracting vertical planar features to correct the vehicle pose in 2D. Bauer [1] proposed a method for the coarse alignment of 3D point clouds using extracted 3D planes that they both are visible in each scan, which leads to reduce the number of unknown transform parameters from six to three. The remained unknown transformation are calculated by an orthogonal rectification process and a simple 2D image matching process. In [52], surfaces are extracted from range images obtained by a rotating laser range finder (LRF) and registered together. A local module determines the correspondences and computes transformations, an a global module detects loop-

closure and distributes uncertainty using an elastic graph.

Other previous approaches using plane features include Pathak group's work [72][71]. In [72], planar features are extracted at each robot pose. These planar features are matched against the features which were seen in prior poses to find correspondences. A new algorithm, called Minimally Uncertain Maximal Consensus (MUMC) for determining the unknown plane correspondences by maximizing geometric consistency by minimizing the uncertainty volume in configuration space. This technique does not make use of any odometry, which enables it to associate planes in successive scans more efficient than typical RANSAC techniques. The authors then computed the least squares rotation and translation which bring the associated planes into alignment. The rotation and translation are used to build a pose graph which is optimized.

In [47], the line and plane features were used together. The authors introduced a framework to integrate point, line and plane features together, and, comparing the integrated method with algorithms that use such features separately, they found that the integrated method is much more stable than the others. In [84], an automated feature-based registration algorithm which searches corresponding pairwise lines and planes in 3D point cloud was presented. Then the registration was embedded in a pose-graph implementation for SLAM. Kohlhepp *et al.* [51] [52] [24] proposed 3D environment mapping approaches using planar features. In Harati's dissertation [40], a hierarchy of geometrical features, adapted to indoor conditions was developed. Besides the line and plane features, it also includes orthogonal corners and cuboids as higher level landmarks which are then constructed to capture certain joint configurations of the base features. Its main idea is to obtain joint associations for planar patches which would be much more robust than individually established bindings.

2.3.1 Iterative Closest Point (ICP)

Iterative closest point (ICP) algorithm is the most popular method for registering geometric 3D point clouds in a common coordinate system. The goal of ICP is to find the rigid homogeneous transformation $\boldsymbol{T}$, consisting in rotation $\boldsymbol{R}$ and translation $\boldsymbol{t}$, that best aligns a cloud of scene points $\mathcal{P}_l$ with the point cloud $\mathcal{P}_r$ presenting the same scene in other views.

The alignment process consists in minimizing an error metric based on the distance between pairs of corresponding points. Usually, the Euclidian distance between corresponding points is adopted as the error metric. At each iteration, the algorithm computes correspondences by finding closest points through the given

initial estimation, and finding the best translation and rotation that minimizes an error metric based on the distance between them. As mentioned above, an initial estimate is required. In fact, a good initial estimation is essential to avoid running into a local minimum position for ICP registration method, that means the overlapping region of registered point clouds should be large. Algorithm 1 is a pseudo-code description of the ICP algorithm for estimating of the aligning rigid transformation between point cloud $\mathcal{P}_l$ and point cloud $\mathcal{P}_r$.

Algorithm 1 $\boldsymbol{T}$ =*transEstimationICP*$(\mathcal{P}_l, \mathcal{P}_r)$

Input: Point clouds $\mathcal{P}_l$ and $\mathcal{P}_r$ with overlapping region.
An initial estimation $\boldsymbol{T}_0$ which transform $\mathcal{P}_l$ to the reference frame of $\mathcal{P}_r$
The pre-defined maximum number of ICP iterations N_t
The allowed maximum difference between two transformation matrices Δd_t
Output: optimum transformation $\boldsymbol{T} = [\boldsymbol{R}\ \boldsymbol{t}]$

1. Select reference points in $\mathcal{P}_l$, then obtain reference points set $\mathcal{Q}_l = \{\boldsymbol{p}_{l,1}, \boldsymbol{p}_{l,2}, \ldots, \boldsymbol{p}_{l,N_q}\}$
2. **for** $i = 1$ to N_t **do**
3. **for** $j = 1$ to N_q **do**
4. $\hat{\boldsymbol{p}}_{l,i} = \boldsymbol{T}_{i-1}\boldsymbol{p}_{l,j}$
5. $\boldsymbol{p}_{r,kj} = \arg\min(||\boldsymbol{p}_{r,k} - \hat{\boldsymbol{p}}_{l,j}||_2^2) \quad \boldsymbol{p}_{r,k} \in \mathcal{P}_r, k = 1, 2, \ldots N_r$
6. $\triangleright$ find the closest point of $\hat{\boldsymbol{p}}_{l,j}$ in $\mathcal{P}_r$
7. **end for**
8. $\boldsymbol{T}_i = \arg\min(\sum_{j=1}^{N_q} ||\boldsymbol{T}\boldsymbol{p}_{l,j} - \boldsymbol{p}_{r,kj}||_2^2)$
9. $\triangleright$ get a new optimum transformation
10. **If** $(||\boldsymbol{T}_i - \boldsymbol{T}_{i-1}||_2 <= \Delta d_t)$ **then**
11. $\triangleright$ check the difference between these two transformation is little enough
12. break;
13. **end if**
14. $i \leftarrow i + 1$
15. **end for**
16. $\boldsymbol{T} \leftarrow \boldsymbol{T}_i$ $\triangleright$ write to the output

2.3.2 SAmple Consensus Initial Alignment (SAC-IA)

SAmple Consensus Initial Alignment (SAC-IA) for registration of 3D point clouds was introduced by Rusu *et al.* [76]. The key element in this algorithm is a new representation, Fast Point Feature Histogram (FPFH), for the target point cloud. FPFH

is a variant descriptor of Point Feature Histogram (PFH) [77] [80] by reordering the order set. It uses a histogram to describe the local geometry around a point $\boldsymbol{p}$ for 3D point cloud datasets.

The SAC-IA algorithm works by applying the following schemes:

1. select sample points from the source point cloud.

2. given a sample point of the source point cloud, instead of searching its matching points directly, the algorithm finds a list of points in the target point clouds whose FPFH features are similar to the sample point's. A random point in the obtained list is considered as the sample point's correspondence.

3. compute the rigid transformation defined by the sample points and their correspondences and compute an error metric, defined by the Huber loss function, for the point cloud that computes the quality of the transformation.

These three steps are repeated, and the transformation that produced the best error metric is stored and used to roughly align the partial views. Finally, a non-linear local optimization is applied to get the final transform result, using a Levenberg-Marquardt algorithm.

From the above description it can be known that, different to ICP, SAC-IA with FPFH does not need an initial alignment of the data samples. Thus, it is a global registration method.

2.4 State-of-the-art of SLAM Back-end

During the past few years, SLAM based on filtering techniques, as EKF, particle filters and information filters, were popular. Recently we have observed a change of paradigms in the SLAM literature. The focus of SLAM research has shifted to optimization-based approaches that have been found to be more efficient, accurate and stable than solutions based on filtering algorithms. A number of optimization based SLAM back-ends are readily available to the SLAM researchers as open source libraries: TreeMap [28], TORO [36], iSAM [49] and very recently Sparse Pose Adjustment [57], HOG-Man [34], iSAM2 [48], and g^{2}o [60].

The first paper that proposed an efficient solution to the full SLAM problem was [64]. The authors explained a technique they called "consistent pose estimation" and applied it to indoor SLAM using laser range finders. The seminal paper represents the SLAM problem using a graph structure.

However, it took several years to make this formulation popular due to the comparably high complexity of solving the error minimization problem using standard

techniques. In 2004, Konolige [53] developed the idea further. He pointed out that the sparse structure is inherent to the full SLAM problem and proposed a preconditioned gradient technique to solve it. GraphSLAM [89] proposed a scheme to reduce the number of variables involved in the SLAM problem by collapsing the constraints between robot poses and landmarks into pose-pose relations. Then a similar approach was presented in [26].

Another alternative view to the back-end problem is considering the spring-mass model in physics. In this view, the nodes are regarded as masses and the constraints as springs connected to the masses. The solution to the mapping problem is computed using an iterative technique, in which the overall system is allowed to 'relax' into the lowest energy state. Relaxation techniques, such as Gauss-Seidel have been presented for obtaining the global optimum configuration of the poses. An early work to use relaxation for the mapping problem was presented in Howard et al [46]. Subsequently a Gauss-Seidel relaxation was proposed in [20] to minimize the error in the network of constraints. Then a variant of Gauss-Seidel relaxation, named multi-level relaxation (MLR) that applies the relaxation at different levels of resolution, was introduced by Frese *et al.*[27].

Recently, Olson *et al.* [70], introduced a stochastic gradient descent approach to further increase efficiency and solve pose graphs despite large initial errors. Later, Grisetti *et al.* [37], extended this approach, by applying a tree-based parameterization, towards non-flat environments with their system called TORO.

Popular solutions for the back-end problem that minimizes the cost function by the given constraints are iterative approaches. They can be processed either by correcting all poses all at once or updating parts of the network incrementally. Recently, in [12], the authors noted that it is possible to compute an accurate linear approximation of the optimum solution under the assumption that the relative orientation and translation are independent. In the following, depending on the techniques used, the optimization approaches are classified into two groups.

Nonlinear Optimization approaches

Nonlinear least squares optimization was used in an approach called $\sqrt{\text{SAM}}$ [19] and its recent incremental enhancements iSAM [49] and iSAM2 [48]. Another strand of nonlinear approaches that explicitly exploits the sparse structure inherent in the SLAM problem was opened by sparse pose adjustment [57]. Considering that, due to the involved rotations, SLAM cannot be correctly modeled using flat, Euclidean spaces, HOG-MAN [34] proposed a manifold approach that proved to outperform the simpler methods operating in Euclidean space. Herzberg *et al.* [43] and Wagner *et al.* [91] developed that manifold approach further and extended

it to general sensor fusion and calibration problems. Combing the insights and learned lessons from HOG-MAN and sparse pose alignment, the publicly available system g^{2}o [60] can be seen as the state-of-the-art approach to solve large-scale SLAM problems containing several (up to 10k) variables (poses, landmarks) and constraints (observations, loop closings) in a matter of seconds on standard hardware.

Kümmerle *et al.* [59] demonstrated the versatility of the g^{2}o framework by extending the state space and adding system parameters that might change over time. In their first experiments, the wheel diameters of a robot were estimated together with the trajectory and the map, leading to simultaneous calibration, localization, and mapping.

Linear Approximations and Closed-Form Solutions

The most recent development in optimization-based SLAM are linear approximations of the SLAM problem that lead to closed-form solutions [12][11]. Such techniques do not require an initial guess and can be solved in a single step instead of iteratively. The general idea is to separate the estimation of orientation and location. The reason for this approach is that an iterative solution is necessary mainly due to the nonlinearities introduced by the orientations. By estimating both quantities separately, the problem can be divided into two linear problems. And the work is extended to [13], where the hypothesis on the structure of measurement covariance is relaxed. Experiments on real and simulated datasets confirmed the validity of the algorithm. The comparison between this algorithm with other state-of-the-art algorithms, for instance, Gauss Newton, g^{2}o, TORO, showed that it has an accuracy which is comparable to other approaches, while it is faster. Further advances may be expected in this area in the future.

2.5 Summary

Graph-based SLAM has recently emerged as a well assessed strategy for the 3D mapping problem. In the context of graph-based SLAM, it typically focus on two problems. The first one is to identify the constraints based on sensor data, often referred to as the SLAM front-end. The second one is to correct the poses of the robot to obtain a consistent map of the environment given the constraints, often referred to as the SLAM back-end.

In this Chapter, a brief theory background of graph-based SLAM was introduced. Afterwards, we presented a review of the two problems of the graph-based

SLAM, front-end and back-end respectively.

Chapter 3

Feature Extraction

As mention before, 3D mapping were successfully addressed by scan-matching approaches. The scan-matching can be implemented in the same level as the raw data points obtained from the Microsoft Kinect. Single points may be treated as orientation-less features which are less certain and less distinguishable, resulting to a big uncertainty during the matching process. While the mentioned disadvantages can be compensated by matching a large amount of points (for example using ICP algorithm), this increases the computation time. Thus, a more efficient alternative is to use features which are less frequent, but more informative, certain and unique. As a result the whole scan-matching procedure will run more efficiently. Meanwhile, it will form a conceptual point of view, providing a more compact, abstract and structurally informative representations which greatly enhance the robot interaction with humans.

Therefore, obtaining abstractions over raw sensory data samples is an important capability of a mobile robot and a key issue for 3D registration. Depending on the working environment of the mobile robot and obtained sensory data samples, different features are implemented. In this work, the indoor environments are considered. As we all know, in indoor environments, several structures like doors, walls, tables, ground floor, etc., can be modeled as planar surface patches, which are parallel or perpendicular to each other. Therefore, planar patches have been found to be a good feature for 3D visual SLAM, while also being a quite good representation for the final 3D map. Fortunately, with the availability of Microsoft Kinect 3D sensor, a dense 3D point cloud along with a color image, representing the surrounding environment of robot, is achievable at high frequency. In later processing steps, plane features will be used to gather higher level features for estimating the robot poses and representing the 3D map of the robot's environment. A combination of several orthogonal planar surface segments may form a room, a corridor and so on. The main disadvantage of a feature-based approach is that depending on the feature type

used and feature accuracy, a big effort has to be summoned in order to extract this feature in a robust way and accurately. This is especially difficult with noisy sensor data containing irregularities and outliers.

Orthogonal corners are another kind of quite reasonable choices for representation of the structured indoor environment. Compared with planar features, orthogonal corner feature is more distinguishable and robust. It encodes the relationship between its component planes. In our work, it is also used to track the robot poses. In this Chapter, extraction of planar patches and orthogonal corners are discussed as features, and the extraction procedure is described.

3.1 Point Cloud Pre-processing

For the convenience of explanation, we will refer to a collection of 3D points as a point cloud structure $\mathcal{P}$. Point clouds provide discrete, but meaningful representations of the surrounding world. Without any loss of generality, the $\{x_i, y_i, z_i\}$ coordinates of any point $\boldsymbol{p}_i \in \mathcal{P}$ are given with respect to a fixed coordinate system, usually having its origin at the sensing device used to acquire the data. In our work, a Microsoft Kinect is used as the sensing device. This means that each point $\boldsymbol{p}_i$ represents the distance on the three defined coordinate axes from acquisition viewpoint to the surface that includes the sampled points.

The dense point clouds acquired by Microsoft Kinect are noisy and suffer from different error sources, especially discretization effects in depth measurements and the fact that the cameras are calibrated for a certain range. Both effects cause considerable measurement errors in far range. Though the Microsoft Kinect's official distance limit is 3.5 meters, actually Kinect acquires depth images of points being farther than this distance but there is a decrease of the cloud's accuracy. In addition, for each frame, the Kinect acquires a point cloud with 307200 (640×480) points, corresponding to the dimension of the acquired depth image. In order to enhance the quality of each dense point cloud, meanwhile to keep the overall processing time reasonable, four kinds of fast filtering methods are used to modify the data.

1. Pass through filter: A major disadvantage of the Kinect camera is the increasing depth discretization error for large distances. There are a lot of points whose depth are out of the operational depth range, even further. The points whose depth are out of a determined threshold are considered to be noisy and shaky. Meanwhile, in order to reduce computation time, the points which are out of interested area can be removed. For this reason, the pass through filter is being used here to cut off the points are out of a pre-determined threshold.

2. Statistical outlier removal: For eliminating sparse outliers which caused by measurement errors, a statistical outlier removal filter is used. It is based on the computation of the distribution of point to its neighbors distances in the input dataset. For each point $\boldsymbol{p}$, compute the mean distance from it to all its k neighbors. By assuming that the resulted distance distribution is Gaussian with a mean and a standard deviation, all points whose mean distances are outside an interval $\mu_d \pm \alpha \cdot \sigma_d$ defined by the global distances mean μ_d and standard deviation σ_d can be considered as outliers and trimmed from the dataset. The parameter α controls the width of the interval and acts as a band-stop filter cut-off parameter.

3. Voxelgrid filter: Voxelgrid filter is used for point reduction, so to reduce computational time and memory usage. Moreover, duplicate points can be removed through downsampling. It works as follows: the dense point cloud is divided into a set of tiny 3D boxes (voxels) with a determined width in space, i.e., voxelgrid filter. Then, in each 3D tiny box, all the points present will be approximated with their centroid. The dimension of voxel decides the number of these so-called reduced points. In our report, the voxel with 2 cm dimension is used to downsample the point clouds.

4. MLS-resampling: Moving least square (MLS) algorithm is usually used to reconstruct the surface and remove the data irregularities, which are caused by small distance measurement errors and are very hard to remove using statistical outlier removal filter. It provides a reconstruct surface for a given set of points by interpolating high order polynomials between the surrounding local neighbors. Smoothing and resampling a noisy point cloud allows to obtain more accurate estimation of surface normals and curvatures, which are very important to further point cloud processing, such as segmenting and clustering. Also for a smoothed and resampled point cloud, it is easy and accurate to segment and cluster the points which belong to a plane using RANSAC plane estimator, which will be discussed later.

Figure 3.2 presents the filtering results of a scene point cloud $\mathcal{P}$, which is shown in Figure 3.1.

3.2 Planar Surfaces Extraction

As mentioned before, for indoor environments, planar surfaces and orthogonal corners are quite good choices for presenting the main structures. In this context, the

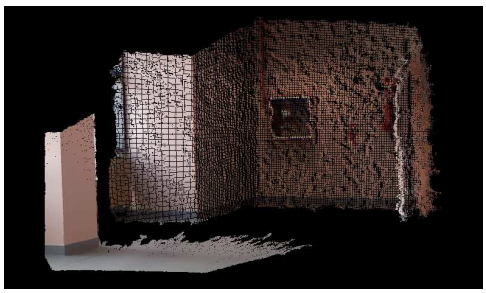
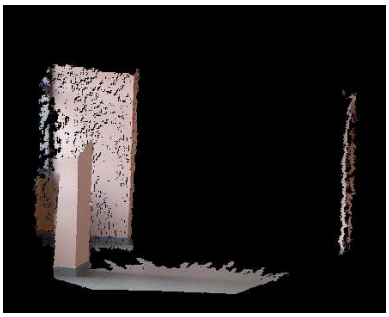
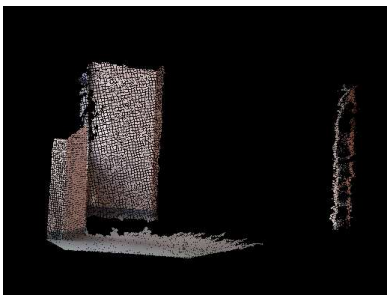


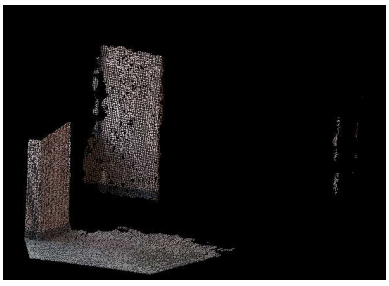
Figure 3.1: An original point cloud of the DAUIN corridor



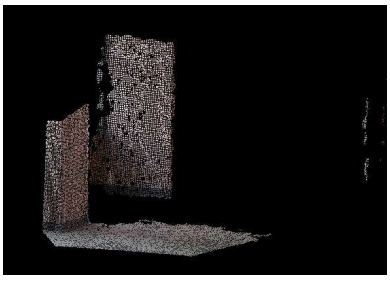
(a) point cloud after pass through filtering



(b) point cloud after voxelgrid downsampling



(c) point cloud after outlier removing



(d) point cloud after MLS resampling

Figure 3.2: Point cloud filtering and reduction results

planar surfaces are used as the basic features to build the spatial relationships between robot poses and the presentation of the environment of robot. Thus planar surfaces extraction is the first step in trying to make some sense out of sensory raw dense point clouds. This section describes the extraction of planes and orthogonal corners from 3D point clouds acquired by Microsoft Kinect.

In this work, we only consider roughly vertical and horizontal planes.The motivation for this choice comes from the fact that in most indoor engineered environments, major structures, like walls, windows, cupboards etc., can be represented by sets of planes which are either parallel or perpendicular to each other. Actually, ignoring other planes (arbitrary oriented or non-orthogonal) not only does not lead to loss of valuable information during 3D mapping, but also brings robustness on the robot orientation and filter out many dynamic objects.

In mathematics, a planar segment is composed of an infinite plane described in general by the following equation:

$$Ax + By + Cz + D = 0 \tag{3.1}$$

where A, B, C, D are the plane parameters and (x, y, z) the coordinates of a 3D point lying in the plane. And (A, B, C) forms the normal vector $\vec{n}$ of this plane. Since actually three parameters are enough to specify a plane in $\mathbb{R}^3$, the normal vector is usually normalized, i.e., $|\vec{n}| = 1$. The constraint of unit length of normal removes the extra fourth degree of freedom and leaves the other three. This notation has the advantage of having normal vector handy which is very useful for the following procedures. Thus it will be used to represent the extracted plane.

3.2.1 Extraction Algorithms

Two different approaches are widely used for extracting planar segments from point clouds. One is based on RANSAC plane model, and another is using region-growing approach. In the following, these two general algorithms are introduced and compared. Combined to the situation in our work, a suitable one will be selected for the planar surfaces extraction.

RANSAC

RANdom SAmple Consensus (RANSAC) [25] is a method to robustly fit a model into a set of data points that may contain even a large number of outliers. It randomly selects a minimal set of data points for estimating the model parameters. From the random samples, it chooses the one that is best supported by the complete set of

points. As of its general formulation, RANSAC can be easily applied to fit any kind of geometric shape primitive.

Algorithm 2 is a pseudo-code description of the RANSAC algorithm for segmenting a single plane from a point cloud. It is mentioned explicitly as it forms the first part of the whole procedure of planar surfaces extraction presented below. For a predefined number of iterations N_I, the segmentation performance of a plane defined by three randomly chosen vertices $\boldsymbol{p}_1$, $\boldsymbol{p}_2$ and $\boldsymbol{p}_3$ is evaluated by counting the number of points N_P lying within a predefined orthogonal distance τ_r. The plane with the highest number of supporting points N_M is output as the best planar segment P found.

The quality of the resulting segmentation directly depends on the predefined distance threshold τ_r and the chosen number of iterations N_I. The chance to find a correct segmentation increases by augmenting the number of iterations N_I. However, the higher N_I, the slower the algorithm. Hence, a trade-off in speed has to be taken into account to realize good segmentation results. The complexity of the RANSAC algorithm can be expressed as $O(N_I \cdot N_p)$.

Let $p_g \in [0, 1]$ be the probability that a randomly chosen data item is part of a good model and $p_f \in [0, 1]$ be the probability that the algorithm terminates without finding a good segmentation. p_g and p_f are related by $p_f = (1 - p_g^{N_M})^{N_I}$. Here, $N_M = 3$ as three data items are necessary in order to describe a plane. Hence,

$$N_I = \frac{\log(p_f)}{\log(1 - p_g^3)} \tag{3.2}$$

Unfortunately, p_f and p_g are generally not known a priori and change from scene to scene. Therefore an empirical analysis is necessary.

Note that the plane found by this algorithm is not necessarily a connected region, which is caused by the planar segmentation definition taken from [31], which examplifies the shortcomings of this definition. For example, a long corridor wall interrupted by doorways or windows could be represented by a single mathematical plane consisting of several planar patches. Therefore post-processing is necessary to offset these shortcomings and to make the extracted planes approximate estimate of the true surface geometry.

Region-Growing

A region-growing algorithm starts from single entities of an input range like points or planar patches and grows these into larger regions by merging them with matching neighbors. When a certain stopping criteria is reached, e.g., if the approximation error of a planar region exceeds a tolerance threshold, the growing process ends. An

Algorithm 2 P=*planarExtracRansac*($\mathcal{P}_t$)

$\mathcal{P} = \{\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_{N_p}\}$ *input* point cloud composed of N_p 3D points
N_P number of points within the environment of the currently defined plane
N_M found maximum number points in the defined environment of the plane
N_I predefined number of RANSAC iterations
d orthogonal distance to plane
τ_r maximum allowed distance for supporting points
(A, B, C) unit normal of plane
P *output* largest found plane

$N_M \leftarrow 0$
for $i = 1$ to N_I **do**
  randomly select 3 different $\boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{p}_3$ of the input point cloud $\mathcal{P}$
  $(A, B, C, D) \leftarrow$ *detectPlane*$(\boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{p}_3)$, $\triangleright$ plane $P : Ax_i + By_i + Cz_i + D = 0$
  for $j = 1$ to N_p **do**
    $d \leftarrow$ *distanceToPlane*$(\boldsymbol{p}_j, P)$
    if $d \leq \tau_r$ **then**
    $N_C \leftarrow N_C + 1$
    endif
  endfor
  if $N_C > N_M$ **then** $\triangleright$ get plane with maximum number of supporting points
    $N_M = N_C$
    $\hat{P} \leftarrow P$ $\triangleright$ write to the largest plane $\hat{P}$
  end if
end for

example of a region-growing algorithm for planar surfaces detection is in [39]. The algorithm starts by randomly selecting a point $\boldsymbol{p}_1$ of the input point cloud $\mathcal{P}$ and its closest neighbor $\boldsymbol{p}_2$. A candidate point $\boldsymbol{p}_i$ is added to the set of planar points ψ if the minimal distance from ψ to $\boldsymbol{p}_i$ is less than a threshold δ. The point $\boldsymbol{p}_i$ is accepted if, when added to ψ, the average residual is less than a threshold ϵ and the distance between the optimal plane and $\boldsymbol{p}_i$ is less than a threshold γ.

According to the above description of region-growing algorithm, we can know that the selection of seed points is a key issue. However, automatic selection of good seed points is very difficult to achieve. It is a well-studied topic, but it has yet to be solved. Based on this issue, in this work, RANSAC-based approach is chosen to extract the existing planes from the filtered point clouds.

3.2.2 Planar surfaces extraction using RANSAC

Given a 3D point cloud $\mathcal{P}_t$ acquired at time t of the indoor environment, after point filtering and downsampling processes, the well known RANSAC plane model is used to segment out all the horizontal and vertical surfaces such as floors, doors, pillars and walls that are present within it. RANSAC is iteratively executed to extract the largest plane from the full point cloud $\mathcal{P}_t$ until a pre-defined ending condition is met. For each iteration of the RANSAC algorithm, the plane with the largest number of inliers is filtered from the full point cloud and returned. For the convenience of presentation, the extracted plane is denoted as $P_{t,i}$, where t is the index of point cloud sample, and i is the index of extracted plane in point cloud $\mathcal{P}_t$.

For the purpose of our work, only planes that are roughly horizontal and vertical are considered. Thus the extracted plane is tested by the relative relationship between its normal vector $\vec{n}$ and the $\vec{z}_r$ axis of robot reference frame, which points upward. Here it should be mentioned that acquired point clouds are referenced to the Kinect camera's reference frame R_k. Therefore it is necessary to transform the point clouds from R_k to R_r. The transform matrix can be computed by the relative relationship between robot and Kinect reference frame. In our work, all the point clouds are transformed to the robot reference frame when the pre-processing procedure is finished.

The horizontal and vertical surfaces are categorized by their relationships with $\vec{z}_r$, as illustrated by the following equations:

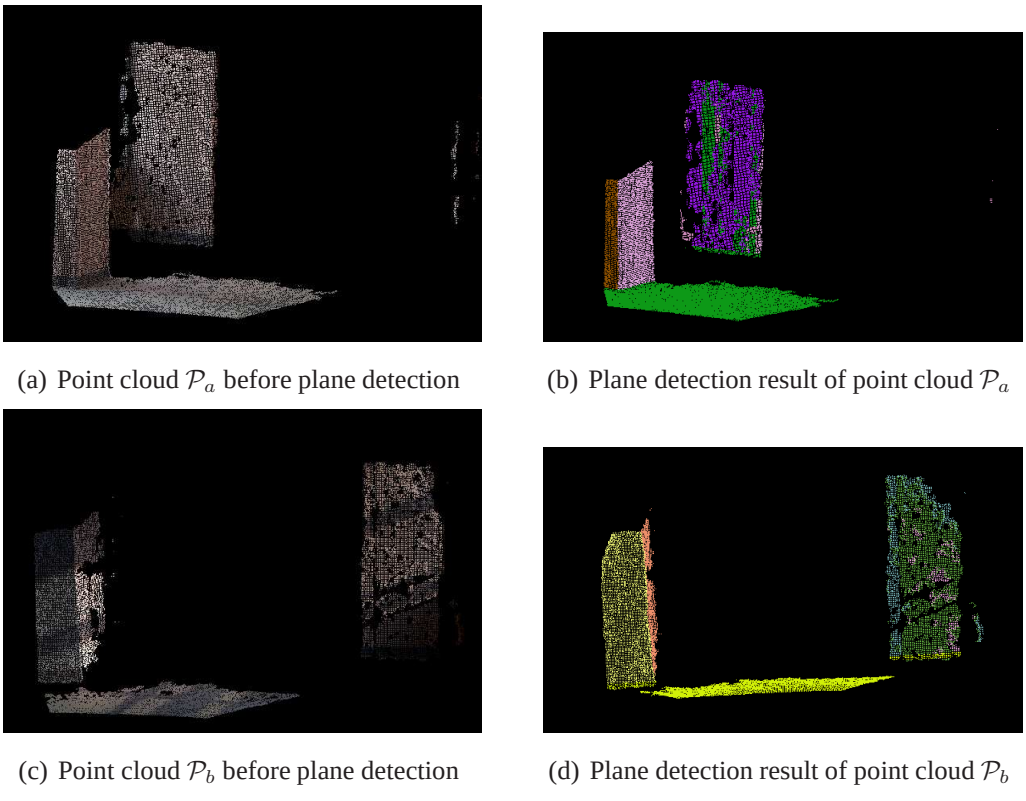
$$P_v = \{P_{t,i} : |\vec{n}^T \cdot \vec{z}_r| < \cos(\frac{\pi}{2} - \phi)\} \tag{3.3}$$

$$P_h = \{P_{t,i} : |\vec{n}^T \cdot \vec{z}_r| > \cos(\phi)\} \tag{3.4}$$

where P_v is a vertical plane, P_h is a horizontal plane, and ϕ is the maximum accept-

able deviation. If the extracted plane $P_{t,i}$ is far from being horizontal or vertical, all the inliers supporting this plane is removed from the whole current point cloud, and then RANSAC is executed again to find the next largest plane. While if it is nearly vertical or horizontal, a distance-based clustering is performed on its points to find large contiguous regions of points within the plane and discard clusters that are too small, as will be discussed explicitly below. Then the supporting points are removed from the point cloud. The same procedure is applied iteratively until no plane with sufficient number of points can be found. A threshold of 1000 was used for this work. This procedure ensures that most of the arbitrary orientated planes are filtered out. Actually, ignoring these kind of planes not only will not lead to a valuable information loss, but it will filter out many useless planes, and simplify the following task to a certain extent.

Compared with other methods that classify the points belonging to a same plane according to their normal vector, the RANSAC is very fast since no reprocessing is required to estimate the normal vector at each point. Figure 3.3 illustrates the plane extraction result using RANSAC in two frames.



(a) Point cloud $\mathcal{P}_a$ before plane detection

(b) Plane detection result of point cloud $\mathcal{P}_a$

(c) Point cloud $\mathcal{P}_b$ before plane detection

(d) Plane detection result of point cloud $\mathcal{P}_b$

Figure 3.3: Two examples of plane detection result. Colors were randomly selected for the planes.

3.2.3 Plane Clustering

RANSAC robust plane estimator is used successfully to find the main horizontal and vertical planes in a scene. However, apparent from Figure 3.3, the plane inliers are fitted to the same mathematical plane but actually on the different sides of the environment, or belong to different physical planes. This is due to the fact that they lie on the same mathematical plane defined by the RANSAC algorithm. Depending on which planar model is randomly created first, the points might belong to one of multiple planes. Obviously, the extracted plane does not reflect the real geometry structure of the scenario and it is not accurate. In order to solve these problems, a distance-based clustering step is performed on extracted planes. Generally, this clustering step serves two purposes: to remove individual points or small clusters of points that fit to the plane but are not part of a large contiguous surface (e.g., a door frame leaning out from the surrounding wall), and to separate multiple surfaces that are coplanar but are in different locations, such as two tables at the same height. Each cluster with a sufficient number of points (a threshold of 500 was used for this work) is saved and will be used for mapping purposes.

The essence of segment and cluster is to group the points with the same properties (e.g. normal, curvature, color) together based on a given measure. In order to achieve the goal, what we need to do is to find a suitable measure which can find an object point cluster and differentiate it from another point cluster at the same time. Usually the measure is the Euclidean or Mahalanobis distance metrics. In our work, the former is used.

For an unorganized point cloud $\mathcal{P}$, a cluster is defined as follows:

Let $O_i = \{\boldsymbol{p}_i \in \mathcal{P}\}$ be a distinct point cluster from $O_j = \{\boldsymbol{p}_j \in \mathcal{P}\}$ if

$$\min \|\boldsymbol{p}_i - \boldsymbol{p}_j\|_2 \geq d_c \tag{3.5}$$

where d_c is a maximum distance threshold. When distance between a set of points O_i and another set of points O_j is larger than this threshold, they are assigned to two different clusters. So the distance threshold is important, which determines the final clustering result is good or not.

A basic algorithm for cluster can be described as follows:

1. for the input point cloud dataset $\mathcal{P}$, create a kd-tree representation. The kd-tree representation is the most used method to find closest points since it is fast to process.

2. set up an empty list of clusters $\mathcal{C}$, and a queue $\mathcal{Q}$ of the points that need to be checked.

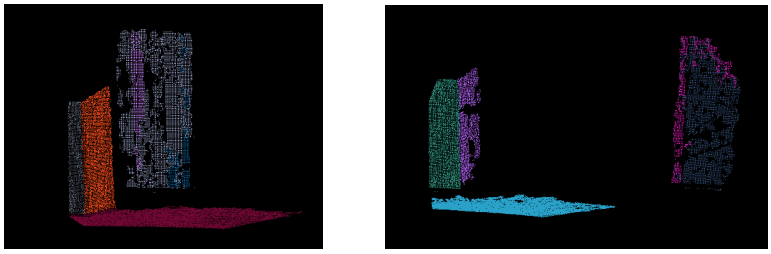
3. then for every point $\boldsymbol{p}_i \in \mathcal{P}$, perform the following steps:

   - add $\boldsymbol{p}_i$ to the current queue $\mathcal{Q}$,
   - for every point $\boldsymbol{p}_i \in \mathcal{Q}$ do,
     - search its neighborhood points set $\mathcal{P}_i^k$ in a given method, such as k-nearest method or radius method;
     - for every neighbor $\boldsymbol{p}_i^k \in \mathcal{P}_i^k$, check if the point has already been processed, and if not add it to $\mathcal{Q}$;
     - when the list of all points in $\mathcal{Q}$ has been processed, add $\mathcal{Q}$ to the list of clusters $\mathcal{C}$, and reset $\mathcal{Q}$ to an empty list.

The algorithm terminates when all the points $\boldsymbol{p}_i \in \mathcal{P}$ have been processed and are now part of the list of point clusters $\mathcal{C}$.

Given a segment plane, using the cluster algorithm to detect whether all the plane points belong to a same object. If the points belong to several clusters, as mentioned before, check each cluster's supporting points number. Then only the clusters which meet the rules are accepted and saved, otherwise it will be removed.This technique can make sure that surfaces with relatively few supporting points such as surfaces that were scanned from far away, will not be detected.

Figure. 3.4 presents the plane clustering results of the above plane detection results shown in Figure 3.3. From the Figure 3.4(a), it can be observed that the pink points in the right side of Figure 3.3(b) are removed, also the yellow points in the bottom part of the right wall in the Figure 3.3(d) are not clustered into the floor surface. At the same time, it should be noted that the small cluster on the utmost right side are deleted since it has few supporting points and its area is too small.



(a) plane clustering result of point cloud $\mathcal{P}_a$

(b) plane clustering result of point cloud $\mathcal{P}_b$

Figure 3.4: Plane clustering results of above Figure 3.3(b) and Figure 3.3(d)

3.2.4 Plane Merging

As discussed above, a clustering step is performed to separate multiple coplanar surfaces such that the points grouped into a plane belongs to a geometric plane. Indeed in some cases the points in a same plane are split into several pieces, i.e., over-segmentation, which means that the number of output planar segments is larger than the number of segments existing in the physical reality. This is mainly caused by noise, occlusion or simply observing different parts of the same plane apart from each other. From Figure 3.4, it can be observed that the back walls are split into two or three pieces, respectively.

To compensate for over-segmentation effect, when two planes are approximately aligned, i.e., they have approximately equal plane coordinates, and are overlapped or near each other, it is desirable to merge them into a larger plane in order to improve the correct matching ratio.

Given a set of planes P_t extracted from point cloud $\mathcal{P}_t$, a basic method comparing all planar segments among themselves has been implemented. For a pair of two plane clusters $P_{t,i}$ and $P_{t,j}$, if the following equations:

$$|\vec{n}_{t,i} \cdot \vec{n}_{t,j}| \geq \cos(\phi_m) \tag{3.6}$$

$$|d_{t,i} - d_{t,j}| \leq \Delta d_m \tag{3.7}$$

$$d(P_{t,j}, P_{t,j}) \leq \Delta d_P \tag{3.8}$$

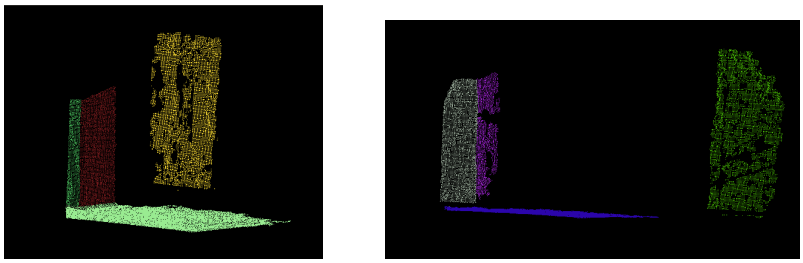
are satisfied, they are considered to be coincident and merging procedure is implemented, where Eq. (3.6) and Eq. (3.7) are presented to determine whether plane clusters $P_{t,i}$ and $P_{t,j}$ are coplanar, while Eq. (3.8) is to check whether they are overlapped or near each other. And ϕ_m, Δd_m, Δd_P are pre-defined maximum acceptable parameters.

Here the overlap is evaluated by finding the neighboring points in a given radius r_m using kd-tree. In order to reduce the computing time, their polygon boundary points are used instead of the entire plane cluster points. For each point, if there are sufficient number of neighbor points in another plane, the point is considered to be near to another plane. If a proper part of points are near to another plane, we assume that these two planes are overlapping or near each other.

After all the planes are refined, small planar surfaces are ignored. This is done by thresholding the patch surface area. For the experiment considered in this thesis, surfaces smaller than 0.25 m^2 are ignored. This technique ensures that surfaces with relatively small area such as surfaces that were scanned from far away will not be detected. In addition, the mathematical parameter of the merged segment will be renewed using RANSAC. Then the refined planes are stored in vertical planes

set P_t^v or horizontal planes set P_t^h according to their classification determined by Eq. (3.4).

Figure 3.5 presents the refined planes, where the planes in a same physical plane are merged together and there is no small planar patches.



(a) Plane merging result of point cloud $\mathcal{P}_a$

(b) Plane merging result of point cloud $\mathcal{P}_b$

Figure 3.5: Plane merging results of Figure 3.4 respectively.

3.3 3D Corners

Toward more robust data association, in this thesis we are interested in understanding how the extracted planes relate with each other, and constructing higher level features based on them. For instance, three orthogonal planes defines a corner, which is more robust and distinguishable in indoor environment. Moreover, a single corner is enough to lock all degrees of freedom in space, since we can associate to a corner both an orientation and a position, while a plane only constraints a distance and an orientation. The idea of grouping basic features to form higher features, which are more distinguishable and less frequent, is fairy general. In our work, three orthogonal planar surfaces defines a 3D orthogonal corner feature. This is glued to the geometrical structures in indoor environment. In long run, corner features can enrich the 3D map with information which help the robot understand its surrounding space.

Although the idea of grouping three planes using 3D corners is fairly general, in our work, we only consider three orthogonal planes, which are common in indoor environment. Usually, an orthogonal corner is constructed by two walls and the floor or the roof. Given a set of three planes $\{P_{t,i}, P_{t,j}, P_{t,k}\}$, a corner $\mathrm{C}_{t,i}$ is constructed if all of the following conditions are met.

1. Every two planes among the set of three planes, are perpendicular to each

other,i.e.

$$\vec{n}_i^{\top} \cdot \vec{n}_j \simeq 0 \tag{3.9}$$

$$\vec{n}_i^{\top} \cdot \vec{n}_k \simeq 0 \tag{3.10}$$

$$\vec{n}_j^{\top} \cdot \vec{n}_k \simeq 0 \tag{3.11}$$

2. The intersection point $\boldsymbol{p}_{ijk}$ lies approximately inside the boundary of its parent planes $P_{t,i}$, $P_{t,j}$ and $P_{t,k}$ or its distance to the nearest boundary point is smaller than a predefined value δ_c.

   Mathematically, this can be expressed as equation

$$d(\boldsymbol{p}_{ijk}, \boldsymbol{B}_{t,i}) \leq \delta_c \tag{3.12}$$

$$d(\boldsymbol{p}_{ijk}, \boldsymbol{B}_{t,j}) \leq \delta_c \tag{3.13}$$

$$d(\boldsymbol{p}_{ijk}, \boldsymbol{B}_{t,k}) \leq \delta_c \tag{3.14}$$

   where $\boldsymbol{B}_{t,i}$, $\boldsymbol{B}_{t,j}$,$\boldsymbol{B}_{t,k}$ are the boundary points set of the plane $P_{t,i}$,$P_{t,j}$ and $P_{t,k}$, respectively.

Each corner is represented by its position in space $\boldsymbol{p}_C$, three normal vectors, $\vec{n}_1$, $\vec{n}_2$, and $\vec{n}_3$, corresponding to its parent planes which denote its orientation. And its position is determined by the intersection lines formed by its parent planes. Thus, to get the position information of orthogonal corners, firstly the intersection line of two meeting vertical planes are detected, then its intersection point with the third horizontal plane is considered as the position of detected corner.

3.3.1 Intersection Line between Two Planes

In order to get the intersection line of two orthogonal planes, we follow the technique in [58]. The line of two planes intersection is normally represented as a point on the line $\boldsymbol{p} = (x, y, z)$ and a direction vector $\vec{n} = (n_x, n_y, n_z)$ emanating from this point. While the direction vector $\vec{n}$, can be computed as the cross product of the two normal vectors of its parent planes. The point on the line, $\boldsymbol{p}$, can legitimately be any point on the line. Mathematically it does not matter what the point is, as long as its on the line. Here we define the point is the point that is closest to the origin $\boldsymbol{p}_o$, which will give a canonical representation of the line.

According to plane extraction section, the extracted plane model can be presented as $Ax + By + Cz + D = 0$,where normal vector is $\vec{n} = (A, B, C)$. Given two planes with normal vectors $\vec{n}_1 = (A_1, B_1, C_1)$ and $\vec{n}_2 = (A_2, B_2, C_2)$, and points on the two planes $\boldsymbol{p}_1 = (x_1, y_1, z_1)$ and $\boldsymbol{p}_2 = (x_2, y_2, z_2)$. The direction

vector of the intersection line is the cross product of the two normal vectors, i.e., $\vec{n} = \vec{n}_1 \times \vec{n}_2$.

Then we will compute the point on the line $\boldsymbol{p}$. Since the point $\boldsymbol{p}$ must be on both planes, so we have two constraints:

$$(\boldsymbol{p} - \boldsymbol{p}_1) \cdot \vec{n}_1 = 0 \tag{3.15}$$

$$(\boldsymbol{p} - \boldsymbol{p}_2) \cdot \vec{n}_2 = 0 \tag{3.16}$$

The point $\boldsymbol{p}$ should also be as close as possible to the origin point $\boldsymbol{p}_o = (x_o, y_o, z_o)$, and the distances between two points is expected to be minimized. The distance is

$$\|\boldsymbol{p} - \boldsymbol{p}_o\|^2 = (x - x_o)^2 + (y - y_o)^2 + (z - z_o)^2 \tag{3.17}$$

This is a problem which can be solved with Lagrange multipliers, with one objective function (3.17) and two constraints Eq. (3.4) and Eq. (3.15) and (3.16). The function w containing the constraints and objective function is

$$\begin{aligned} w \quad &= \|\boldsymbol{p} - \boldsymbol{p}_o\|^2 + \lambda(\boldsymbol{p} - \boldsymbol{p}_1) \cdot \vec{n}_1 + \mu(\boldsymbol{p} - \vec{p}_2) \cdot \vec{n}_2 \\ &= (x - x_o)^2 + (y - y_o)^2 + (z - z_o)^2 + \\ &\quad \lambda x n_{1x} + \lambda y n_{1y} + \lambda z n_{1z} - \lambda \boldsymbol{p}_1 \cdot \vec{n}_1 + \\ &\quad \mu x n_{2x} + \mu y n_{2y} + \mu z n_{2z} - \mu \boldsymbol{p}_2 \cdot \vec{n}_2 \end{aligned} \tag{3.18}$$

where λ and μ are the two Lagrange multipliers. Then we get the Lagrange multipliers by computing the partial derivatives and setting them to zero. And the final equations in matrix form are

$$\begin{bmatrix} 2 & 0 & 0 & n_{1x} & n_{1x} \\ 0 & 2 & 0 & n_{1y} & n_{1y} \\ 0 & 0 & 2 & n_{1z} & n_{1z} \\ n_{1x} & n_{1y} & n_{1z} & 0 & 0 \\ n_{1x} & n_{1y} & n_{1z} & 0 & 0 \end{bmatrix} \begin{pmatrix} x \\ y \\ z \\ \lambda \\ \mu \end{pmatrix} = \begin{pmatrix} 2x_o \\ 2y_o \\ 2z_o \\ \boldsymbol{p}_1 \cdot \vec{n}_1 \\ \boldsymbol{p}_2 \cdot \vec{n}_2 \end{pmatrix} \tag{3.19}$$

Solving this matrix equation, we can get the unknown vector (x, y, z, λ, μ), so we can get the point $\boldsymbol{p}$ on the line closest to $\boldsymbol{p}_o$.

3.3.2 Corner Detection

A corner is constructed by three sets of planes in different directions. Here the 3D right angle corner, which are intersections of three orthogonal planes, are only considered to be used as high level features for registration and robot localization. Since in our case, the experimental environment is an indoor environment, and because

the accuracy limitation of the Kinect, the points which are very far away from the Kinect with distance along $\vec{z}_r$ axis exceeding a threshold are filtered, most of the corners are constructed by two vertical walls and floor. Based on this, we assume the floor plane is infinite. Actually, only part of the floor can be detected by Kinect.

From the above section, the line of two vertical walls can be obtained. So corners' position information will be determined by the intersection point of the obtained intersection line and floor plane. In the following Figure 3.6, the detected corners in six frames are shown, and the white lines are the normal vectors of constructing planes.

3.4 Summary

In this Chapter, we have presented the planar surfaces features extraction and 3D corners features detection from raw point clouds, leading to more efficient SLAM while at the same time more compact and structurally informative representations of final 3D map which greatly enhance the robot interaction with its environment. The popular RANSAC plane model is used for extracting the largest plane from the raw point clouds. The reported experiments in the context of plane extraction showed the effectiveness and robustness of RANSAC plane model extraction. While since the detected largest plane by RANSAC plane is extracted from a mathematical view, it is possible that points in a same mathematical plane but belong to different geometric object, e.g., two tables at a same height. As mentioned before, the quality of plane segments extracted affects the plane correspondences matching step and the pose registration step. After that, the extracted planes are refined by a distance-based clustering step and a merging step. It should be mentioned that only the roughly vertical planes are saved and delivered to the plane matching part, which will be discussed in next chapter. In addition, the raw point clouds are pre-possessed by several filters to remove noisy points and decrease the computing time.

Meanwhile 3D corners are detected based on the extracted planes. Here only the orthogonal corners, which are common in most of indoor environments, are considered, i.e., three intersecting planes which are perpendicular to each other form a 3D corner. 3D orthogonal corners are more distinguishable and they represent higher features since they encode the relationship between planes. The use of 3D corners improves the accuracy and robustness of data association. The experiment results shows that 3D corners in the obtained point clouds can be detected effectively.

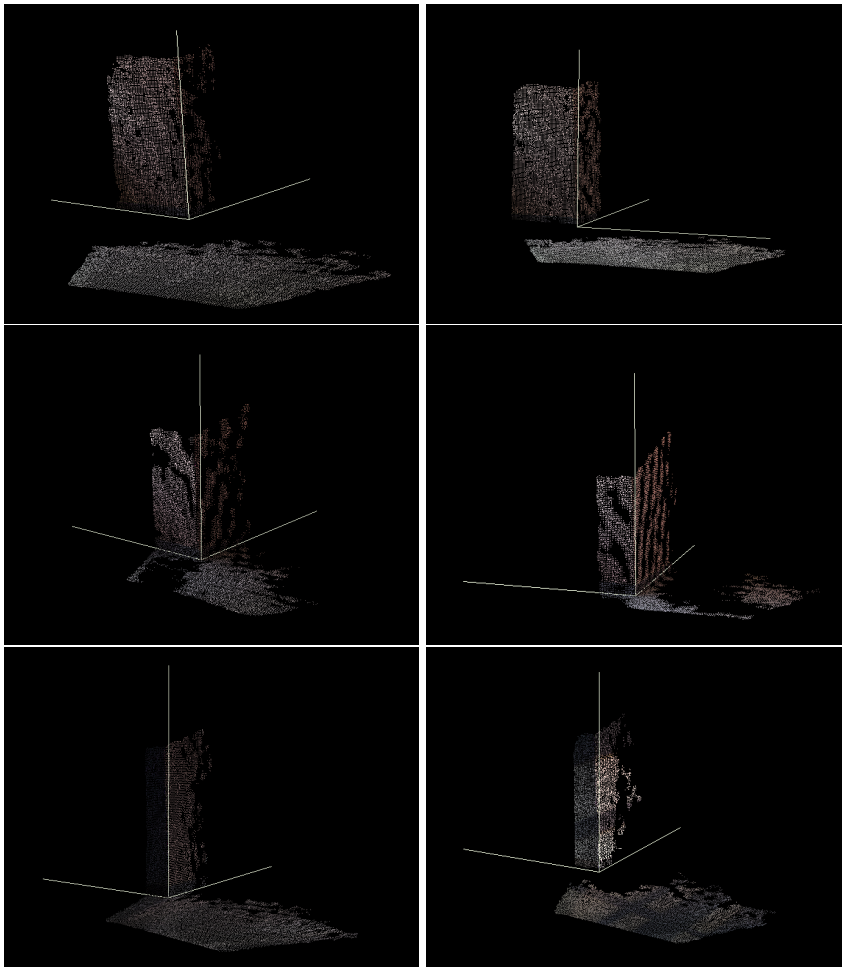


Figure 3.6: Detected corners in 6 frames

Chapter 4

SLAM Front-end

In Chapter 3 the obtained point clouds are processed, in which planes and orthogonal corners features are extracted, providing the necessary material for implementation of a full SLAM loop. As mentioned before, graph-based SLAM is divided into two problems: 1) extracts spatial relations between individual observations. This is referred as the SLAM front-end. 2) optimizes the poses of these observations in a so-called pose graph and with respect to a non-linear error function. This is referred to as the SLAM back-end. In this Chapter, we focus on the front-end part, and two major contributions of this thesis are explained: estimating the pose changes by aligning the corresponding sets of planar surface segments and orthogonal corners, while the corresponding relationship between planes and orthogonal corners in two scans is determined by a plane matching algorithm and a corner matching algorithm respectively.

First a brief introduction to the SLAM front-end problem is presented. Then correspondence problem is introduced, which is one of the most critical problems in feature-based SLAM. It is the problem of finding features in scans taken from different locations that correspond to the same physical entity. In our case, planes and orthogonal corners are considered as the features to be matched. Since the experimental environment is flat almost everywhere with the exception of several small ramps, it is believed that it is suitable to only consider vertical planes in the matching procedure.

How to recognize corresponding planes in different frames robustly is difficult, since wrong matches will result in a big divergence in the trajectory of robot. In this Chapter, a new plane matching algorithm is presented and explained in detail. In order to validate the correctness and robustness of the presented plane matching algorithm, in Chapter 6 pose registration experiments are carried out by using the presented algorithm, ICP and SACIA respectively, where ICP and SACIA are used as baseline algorithms for registration. This Chapter ends with a brief summary

which highlights the important points in few sentences.

4.1 Introduction

In our work, the robot, only equipped with a Microsoft Kinect and wheel encoders, moves in planar indoor environments. The Microsoft Kinect is used for collecting 3D point clouds of the explored environment, while the wheel encoders provides an initial estimation of the displacement between two consecutive scans.

For denoting frames and relative transforms, let us call $\mathcal{P}_j$ the j-th point cloud and the $\mathcal{F}_j$ associated frame from which the point cloud was observed. If the robot, more precisely, the sensor mounted on the robot, moves from frame $\mathcal{F}_j$ to frame $\mathcal{F}_k$, i.e., it undergoes a rotation by $\boldsymbol{R}_k^j$ and a translation by $\boldsymbol{t}_k^j$ between frames $\mathcal{F}_j$ and $\mathcal{F}_k$. Usually, $\mathcal{F}_j$ and $\mathcal{F}_k$ are successive frames, but they may be nonsuccessive, for example during loop closing. The front-end seeks to determine the most likely transformations $\{\boldsymbol{R}_k^j, \boldsymbol{t}_k^j\}$, i.e., constraints between poses from an observation, and then to construct a pose graph that are the basis for the optimization approaches.

Suppose two Cartesian coordinates $\boldsymbol{p}_j$ and $\boldsymbol{p}_k$ of the same physical point observed from the two frames $\mathcal{F}_j$ and $\mathcal{F}_k$ respectively, they are related by

$$\boldsymbol{p}_j = \boldsymbol{R}_k^j \boldsymbol{p}_k + \boldsymbol{t}_k^j \tag{4.1}$$

For odometric edges (i, j) the frames $\mathcal{F}_i$ and $\mathcal{F}_j$ correspond to successive poses assumed by the robot, while for a loop closing, they are non-successive.

Eq. (4.1) is only about point transformation, such as point-to-point (P-P) ICP. It works with the points directly and hence does not assume specific structure in the environment. However, this algorithm is computationally expensive and slow for large point clouds of the order of 10^4-10^6 points. Meanwhile, it suffers from premature convergence to local minima, especially when the overlap between view samples is not large.

If the environment where robot explores has some structures, e.g., in indoor environments, main structures, like doors, walls, tables floors, etc., are made up of many planar surface patches, which are parallel or perpendicular to each other, then scan-matching based on plane segments offers many advantage in terms of computational efficiency and an increase in data association robustness. Furthermore, a map based on plane segments requires few storage memory and is easy to visualize.

Thus in this thesis, planar surface-patches are utilized as the basic features in the front-end part. Our approach falls into the category of estimating relative poses based on the correspondences between large 3D surface-patches extracted from two registered scans.

4.2 Correspondence Problem

Using extracted features to solve geometric estimation problems induces a data-association problem, also known as the correspondence problem. It is considered as one of the most critical problem [88] of SLAM. It is a problem of finding features in scans taken from different locations that correspond to the same physical entity. The higher and the differentiability of used features, the better is the obtained data association performance. Additionally, computational complexity can be further reduced if features are distinguishable, even partly, by restricting the search space to similar candidates. Abstraction levels range from geometric features like points, lines or planes to semantically more significant features combining laser and vision information for high distinctiveness [61].

Different approaches to correspondence problem is classified into two main categories in [93]: discrete matching and iterative alignment. The first one covers the approaches that explore the discrete search space of potential correspondences, while the second category are about the approaches that pose correspondence determination as the problem of searching for the alignment which lines up current observation with the previous one, or the built map. This work is aimed at finding a reasonably fast and accurate matching algorithm to determine correspondences between planar surfaces extracted in consecutive views, and then estimate the relative roto-translation between these two views. As already mentioned, the use of more distinctive features helps to improve the performance of correspondence problem. Orthogonal corners constructed by three planes are considered as higher level features. The establishment of correspondences between corners are discussed too.

Statistical decision is commonly used to measure the difference between different features [81]. In loose words, this means a metric is needed to compare different features quantitatively, taking into account uncertainty information. The Mahalanobis distance [66] d_M is such a metric and is defined as follows:

$$d_M(\boldsymbol{V}_x) = \sqrt{(\boldsymbol{V}_x - \mu)^T \boldsymbol{C}^{-1} (\boldsymbol{V}_x - \mu)} \tag{4.2}$$

d_M is the Mahalanobis distance of a random vector $\boldsymbol{V}_x$ to a multivariate normal distribution with mean μ and covariance matrix $\boldsymbol{C}$. It can also be defined as a dissimilarity measure between two random vectors $\boldsymbol{V}_x$ and $\boldsymbol{V}_y$ of the same distribution with covariance matrix $\boldsymbol{C}_M = \boldsymbol{C}_x + \boldsymbol{C}_y$, yielding

$$d_M(\boldsymbol{V}_x, \boldsymbol{V}_y) = \sqrt{(\boldsymbol{V}_x - \boldsymbol{V}_y)^T \boldsymbol{C}_M^{-1} (\boldsymbol{V}_x - \boldsymbol{V}_y)} \tag{4.3}$$

If $\boldsymbol{V}_x$ and $\boldsymbol{V}_y$ are randomly chosen, $d_M(\boldsymbol{V}_x, \boldsymbol{V}_y)^2$ is a χ^2-variable with r degrees of freedom.

To test whether a current observed feature $\boldsymbol{V}_x$ matches a previous observed feature $\boldsymbol{V}_x$, with N_x being the number of components of $\boldsymbol{V}_x$ or the degrees of freedom, the χ^2-hypothesis test can be carried out by evaluating $d_M(\boldsymbol{V}_x, \boldsymbol{V}_y)^2$. The obtained difference $d_M(\boldsymbol{V}_x, \boldsymbol{V}_y)^2$ between them shows in probability how much they can be matched to an identical feature.

The above discussion of correspondence problem is in a probabilistic framework. In practice, not all hypothetical matchings worth to be further considered. Usually the obtained $d_M(\boldsymbol{V}_x, \boldsymbol{V}_y)^2$ is compared with a predetermined threshold, picked from χ^2 tables with corresponding degrees of freedom and required confidence, to check compatibility of the paired features. If the pairing is less probable than the threshold, it is rejected as a potential hypothesis. Otherwise the features are accepted as a pair of corresponding features.

4.3 Plane Matching

In this work, planar segments are used as the basic features in a 3D SLAM framework. The features extraction from raw point clouds obtained by a Microsoft Kinect is described in Chapter 3.

Consider a robot frame $\mathcal{F}_t$, corresponding to the pose of the robot at time t and an indexed set P_t^v of vertical planar patches extracted from the point cloud $\mathcal{P}_t$ associated with the robot frame $\mathcal{F}_t$. We identify the robot frame with three axis: $\vec{\mathrm{z}}_r$ (already introduced in Chapter 3), which is perpendicular to the plane in which the robot moves, $\vec{\mathrm{x}}_r$ heading towards the direction of motion of the robot, and $\vec{\mathrm{y}}_r$ completing the term.

Based on the assumption that the robot is moving on a plane, the robot poses $\boldsymbol{x}_{1:T}$ are presented as 2D transformations in $SE(2)$. Thus, in this context we only consider the extracted vertical or nearly vertical planes. The motivation for this choice is twofold: first, since we are addressing the planar case, horizontal planes do not provide ”strong” constraints on the robot pose; moreover, in an indoor environment large planar patches are most likely to be walls, doors or other vertical surfaces.

4.3.1 Problem Statement

Given two sets of vertical planes which are extracted from two successive views of a 3D Microsoft Kinect sensor rigidly mounted on a mobile robot, in this section, the goal of the plane matching algorithm is to find correspondences between these current observed planes features and the features observed previously, i.e., answer

the question of "which plane is which". From these corresponding planes, we would like to estimate the change in position and orientation of the robot between the measurement samples.

Generally the overlap between these two frames is unknown. For instance, for two consecutive frames $\mathcal{F}_t$ and $\mathcal{F}_{t-1}$, some planes go out of the view while some new planes come into view which were not previously visible. Thus how to robustly detect that plane feature $P_{t,i}$ in current frame $\mathcal{F}_t$ is the same physical plane patch as plane $P_{t-1,j}$ in previous frame $\mathcal{F}_{t-1}$, is a difficult task.

In [24], a comprehensive discussion on finding correspondences between two sets of planar or quadratic patches using attribute-graphs is presented. In it, similarity metrics are formulated based on several attributes like shape-factor, area ratio, curvature-histogram, inter-surface relations etc., and a bounded tree search was performed to give a set of correspondences which maximized the metric. The result is refined using an evolutionary algorithm, which means the computing-time is high. Here apart from planar patches, orthogonal corners are used, which will be discussed latter. The plane matching algorithm is described which maximize the overall geometric and appearance consistency within a search-space to determine potential correspondences between planes. The search-space is pruned using criteria such as size-similarity, agreement with odometry, and appearance-similarity. Then, based on the fact that, the relative rotation between the pairwise registered scans is unique, which means the relative rotations estimated by all the potential corresponding pairs should be same, or much close to each other. Thus a consistent test is applied to discard the wrong potential correspondences according to their similarity indices defined in this work, then the set of resolved correspondences is obtained.

In the following, the plane matching algorithm will be explained thoroughly.

4.3.2 Plane Feature Representation

As mentioned above, only the extracted vertical or roughly vertical planes are used to build the correspondences between two successive views. In principle, it is suitable to project the vertical planes onto the $\vec{x}_r\vec{y}_r$-plane and use their projections, i.e., 2D lines, to represent these vertical planes. Therefore, a vertical plane represented as $Ax + By + Cz + D = 0$ will be a line represented as $Ax + By + D = 0$ in 2D space. Before delving into the representation used for the vertical planar segments in 2D space, a survey of line models used in the literature is presented in the following.

In mathematics, infinite lines in 2D Cartesian space are generally represented

as:

$$A_l x + B_l y + C_l = 0 \tag{4.4}$$

where A_l, B_l and C_l are parameters.

While the polar form is another common way to represent the line:

$$x\cos\alpha + y\sin\alpha - r = 0 \qquad \text{or equivalently} \qquad \rho\cos(\theta_l - \alpha) - r = 0 \tag{4.5}$$

where α and r are the line parameters: $-\pi \leq \alpha \leq \pi$ is the angle between positive $\vec{\mathrm{x}}$ axis and line normal, and $r \geq 0$ is the distance of origin to the line. The parameter ρ equals $\sqrt{x^2 + y^2}$ and θ_l is computed as $\theta_l = \arctan\frac{y}{x}$. The polar form is preferable since it only uses two parameters that is the minimum number required to represent a line.

In order to present the relationship between the robot and extracted vertical plane-sets better, a different 2D line presentation is introduce here. We parametrize the 2D lines in terms of distance d from the robot and the relative angle $\theta^p \in (-\pi, +\pi]$ (both expressed in the robot frame). The orientation θ^p of the projected plane is defined to be the angle between $\vec{y}_r$ axis and the projection line, see Figure 4.1. The projection line is oriented in clockwise direction such that the robot is always on the right-hand side of the plane. The advantage is that the robot can distinguish from which side a plane is observed, and can distinguish the parallel planes in a same scenario, therefore perform more reliable associations. Thus, in the following, a vertical plane P_v in 2D space is represented by a couple of parameters (θ^p, d), where $\theta^p \in (-\pi, +\pi]$ and $d \geq 0$. Actually the case $d = 0$ is unlikely to occur in practice in the robot frame. Since we assume that the origins of robot frame and sensor frame are located at a same point, d has the same value in the sensor frame.

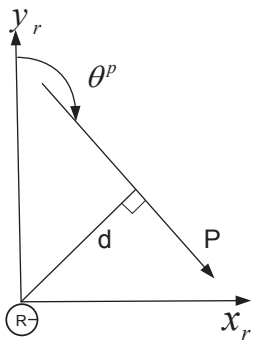


Figure 4.1: Representation of the projection of a vertical plane

4.3.3 Plane Matching Algorithm

For the convenience of explanation, we consider two robot frames: k-th robot frame denoted as $\mathcal{F}_k$ from which an indexed vertical plane-set P_k^v is observed, and another frame $\mathcal{F}_t$ from which the indexed plane-set P_t^v is observed. Note that a vertical plane is presented using parameters (θ^p, d), as described before. Thus P_k^v is an ordered set of plane parameters defined as

$$P_k^v \triangleq \{P_{k,i}(\theta_i^p, d_i), i = 1, 2 \dots N_k\} \tag{4.6}$$

where k is the index of point cloud sample, and i is the index of extracted plane in each point cloud sample. Our goal is to establish correspondences between planes in P_t^v and planes in P_k^v, i.e., which pair $(P_{t,i}, P_{k,j})$ represent the same physical location. These two frames are typically successive for normal registration, but they may also be nonsuccessive, for example if a loop is closed. For simplicity, in this section we only consider the case of plane-sets acquired at consecutive frames, i.e., we consider the case $k = t - 1(t \geq 1)$. And the loop closure detection will be discussed in Chapter 5.

Given a query plane $P_{t,i}$, the ith plane in frame $\mathcal{F}_t$, there are $N_{t-1} + 1$ possible correspondences, if we also include the case plane $P_{t,i}$ is not present in previous frame $\mathcal{F}_{t-1}$. We can naïvely try all of these correspondences, discard wrong matches by using different tests and choose the plane $P_{t-1,j}$ with the maximum overall consistency as the potential correspondence of plane $P_{t,i}$. If the query plane $P_{t,i}$, corresponds to plane $P_{t-1,j}$ in frame $\mathcal{F}_{t-1}$, it will be denoted as $P_{t,i} \leftrightarrow P_{t-1,j}$, abbreviated $i \leftrightarrow j$.

The use of different tests is important in order to discard most of the false correspondences, which are unavoidable in practice. For each plane $P_{t,i}, (i = 1, 2, \cdots N_t)$ in $\mathcal{F}_t$, the following tests are applied one by one to select candidate correspondences for query plane $P_{t,i}$: (1) odometric rotation agreement test; (2) odoemtric translation agreement test; (3) appearance similarity test; (4) size similarity test. A similarity measure is defined to evaluate how good is the selected correspondences.

Odometry rotation agreement test

Since odometry is available, we can use it to choose, among the candidate matches, the ones that meet the rotation agreement with the odometry values. That is based on the assumption that the odometry relative rotation error is bounded in successive frames. To compare the plane $P_{t,i}$ with $P_{t-1,j}$, they have to be transformed into a common reference frame. Since odometry information is given, we can compute

the odometric rotation matrix $\boldsymbol{R}_t^{t-1}$ according to the relative rotation between frame $\mathcal{F}_{t-1}$ and $\mathcal{F}_t$. The query plane $P_{t,i}$ is transformed into its previous reference frame $\mathcal{F}_{t-1}$, and its rotation parameter $\hat{\theta}_i^p$ in reference frame $\mathcal{F}_{t-1}$ is obtained. The two sets of parameters in two frames are related by

$$\vec{n}_{t-1,i} = \boldsymbol{R}_t^{t-1}\vec{n}_{t,i} \tag{4.7}$$

where $\vec{n}_{t,i}$ is normal parameter of query plane $P_{t,i}$, provided by the plane extraction procedure. While $\vec{n}_{t-1,i}$ is its corresponding normal parameter in frame $\mathcal{F}_{t-1}$. Then its renewed 3D normal parameter is transformed to the defined orientation $\hat{\theta}_i^p$ in 2D space according to the technique presented in above section.

Now, we look for all planes in P_{t-1}^v satisfying

$$\|\hat{\theta}_i^p - \theta_j^p\| \leq \Delta\theta_t \tag{4.8}$$

where θ_j^p is the orientation of the j-th plane $P_{t-1,j}$ in P_{t-1}^v and $\Delta\theta_t$ is a fixed threshold. If the orientation θ_j^p of plane is roughly equal to $\hat{\theta}_i^p$, (expressed in frame $\mathcal{F}_{t-1}$), i.e., Eq. (4.8) is satisfied, the j-th plane of P_{t-1}^v is selected as a candidate match for the i-th plane in the set P_t^v, then we add it to the candidate subset P^w while others are rejected as matches do not agree with the odometry rotation.

Odometry Translation Agreement

After test 1 (odometry rotation agreement test), a subset of candidate planes P^w corresponding to the query plane $P_{t,i}$ is obtained. Similar to the previous test, an estimate of the translation $\boldsymbol{t}_t^{t-1}$ is given according to odometry. We can use it to eliminate potential correspondences pairings from P^w which cause a gross disagreement with the odometry values, and keep the correspondences pairings which meet the agreement with the odometry values for the further test. More formally, given a potential correspondence $P_{t,i} \leftrightarrow P_{t-1,j}$, the parameter of distance can be used to select the planes from P^w which are close to query plane $P_{t,i}$. Mathematically, if plane $P_{t-1,j}$ satisfies the following inequality:

$$\|\hat{d}_i - d_j\| \leq \Delta d_t \tag{4.9}$$

where $\hat{d}_i$ and d_j are distances of plane $\hat{P}_{t,i}$ (expressed in frame $\mathcal{F}_{t-1}$) and $P_{t-1,j}$ to the origin in 2D space respectively, and Δd_t is the pre-defined threshold describing how close two planes are required to be, the potential correspondence is considered to pass the odometry translation test, and the plane $P_{t-1,j}$ is added to the subset P^t, while others are rejected.

Texture-based Test

In indoor environment, it is normal that there might be more than one planes with very similar (θ^p, d) values. For instance, when a door is closed, it is parallel to the wall, and usually the door is ahead of the wall by few centimeters, as shown in Figure 4.2. Usually the displacement is less than 10 cm. They are so close that it is inevitable that the parallel door and wall in different frames are wrongly considered as corresponding planes. Disambiguating among them is usually not necessary for relative pose estimation if there are not planes perpendicular to these planes, i.e., they do not form a corner with other planes. However, if they are part of a corner, as shown in Figure 4.2, a wrong match between planes will produce as a result that two corners $C_{t,i}$ and $C_{t-1,j}$ in different frames will be matched to be a same physical corner. The estimated transformation according to this pair of corresponding corners will induce a big error in translation value. In such a case, color feature is considered to be a good option to disambiguate the plane correspondences after implementing the orientation and distance agreement test, since usually the color of the door is different from its surrounding wall's, or the door may have some fancy textures in its dominant planar shape. Meanwhile, the use of color consistent test will make the correspondences finding more reliable.

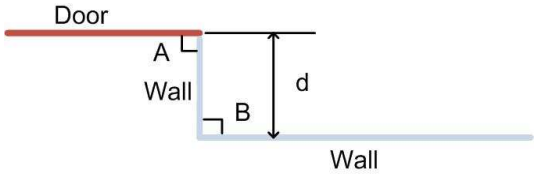


Figure 4.2: Relationship of a door and a wall in indoor environment

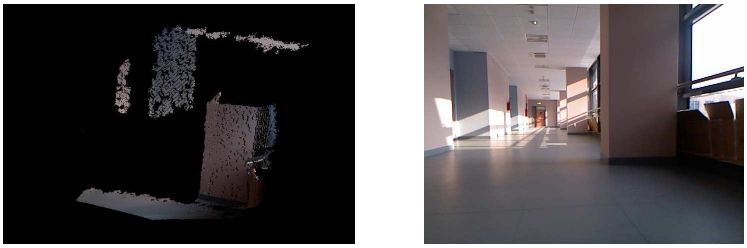
For each point cloud $\mathcal{P}$, besides the points Cartesian coordinate information (x_i, y_i, z_i), it also contains color information associated with every single point. In our case, the color information is presented with RGB color model. Its RGB image I can be obtained by parsing the original point cloud $\mathcal{P}$ according to a certain transformation,

$$p = F(\boldsymbol{p}) \tag{4.10}$$

where p is a pixel in image I, and $\boldsymbol{p}$ denotes a point in $\mathcal{P}$. In this thesis we follow the technique implemented in Point Cloud Library (PCL) [79].

Figure 4.3 shows an example of parsing result. Apparently, the parsing result is quite good and the parsed image can describe the scanned environment accurately.

Given a pair of matching $P_{t,i} \leftrightarrow P_{t-1,j}$ from set P^t, their 2D images $\{I_{t,i}, I_{t-1,j}\}$ will be obtained according to Eq.(4.10). To describe each plane image's information, color histogram is considered. It is one of the frequently used color descriptors



(a) Original point cloud $\mathcal{P}_a$ (b) 2D image I_a of $\mathcal{P}_a$

Figure 4.3: An example of parsing result

that characterizes the color distribution in an image, and it is a flexible construct that can be built from images in various color spaces, whether RGB, HSV or any other color space of any dimension. Here we build the histograms in RGB space.

To compare $I_{t,i}$ and $I_{t-1,j}$ so to check the appearance consistency of potential matching pair $P_{t,i} \leftrightarrow P_{t-1,j}$, the common correlation measure in OpenCV [9] is used to estimate their similarity. Given two color histograms H_i and H_j, the correlation measure is shown in the following equation

$$d_c(H_i, H_j) = \frac{\sum_I (H_i(I) - \bar{H}_i)(H_j(I) - \bar{H}_j)}{\sqrt{\sum_I (H_i(I) - \bar{H}_i)^2 (H_j(I) - \bar{H}_j)^2}} \tag{4.11}$$

where $H_i(I)$ are the bin values of histogram H_i, $\bar{H} = \frac{1}{N}\sum_I H_I$, and N is the total number of histogram bins. From Eq. (4.11), we can get conclusion that a high score represents a better match than a low score. A perfect match is 1 and a maximal mismatch is -1.

For images $I_{t,i}$ and $I_{t-1,j}$, the histograms of three different channels are build and compared respectively, i.e., three correlation value, d_c^r, d_c^g and d_c^b will be obtained. And their product value, i.e., $d_c^r \times d_c^g \times d_c^b$ is considered as the parameter to check the color consistency of candidate matching $P_{t,i} \leftrightarrow P_{t-1,j}$. If the correlation value is larger than a fixed value C_t, it will be chosen as one of the corresponding plane of query plane $P_{t,i}$, then added to the set P^c, otherwise it will be rejected.

Size Similarity Test

In the Appearance similarity Test, color feature is used to distinguish planar patches with very close values (θ^p, d) but different colors. However, in some cases, planar patches not only have close values (θ^p, d), but also similar colors. For instance, a pillar with its neighboring wall. In such a case, one way to distinguish these planes is to consider whether their size are similar, since usually the pillar is smaller than the wall. A size similarity test is presented as Eq. (4.12), which is based on the area

ratio of assumed corresponding planes.

$$\frac{\min(S_i, S_j)}{\max(S_i, S_j)} \geq r_t \tag{4.12}$$

where S_i and S_j denotes the area of plane $P_{t,i}$ and $P_{t-1,j}$ respectively. If the ratio of their area exceeds a pre-defined threshold, plane $P_{t-1,j}$ is accepted as potential corresponding plane of query plane $P_{t,i}$ and added to set P^s otherwise it will be discard.

Similarity Measure

After the four tests presented before, the set P^s may contain non-unique planes, i.e., more than one planes in previous frame $\mathcal{F}_{t-1}$ may be mapped to the query plane $P_{t,i}$. We assume in each plane-set P^v, there are no two plane features belonging to a same geometric plane, as a result of the application of plane clustering and merging. Thus, if we do not consider the case that a query plane $P_{t,i}$ is not present in the previous frame $\mathcal{F}_{t-1}$, only one plane can be mapped to the query plane $P_{t,i}$, i.e., the correspondence is unique.

To solve the uniqueness problem and evaluate the similarity between each pair of candidate matching planes, a similarity metric is defined by evaluating the goodness or reliability of the assumed correspondences. Three factors are used in measuring the similarity between two selected potential corresponding planes. First, their "extent" of agreement with the odometry rotation and translation tests. Moreover, the area factor is considered. The overall similarity of a pair of candidate corresponding planes ie expressed as the weighted sum of different factors as shown in equation:

$$I_s(i,j) = k_s \times \min(S_i, S_j) + k_o \times \frac{1}{|\hat{\theta}_i^p - \theta_j^p|} + k_d \times \frac{1}{|\hat{d}_i - d_j|} \tag{4.13}$$

where k_s, k_o, k_d are three coefficients weighting the importance of corresponding planes areas, orientation and distance agreement with odometry constraint, which is similar to the measure metric proposed in [41].

Contrary to many works in the literature, here there would be no attempt to decide the weighting coefficients from statistical point of view. They are simply chosen according to the empirical knowledge about the odometry error, and the distribution of planes area. Considering the fact that a matching between two large planes is more reliable than a matching constructed by a pair of small ones, and the odometry error is big, in our work, $k_s \times \min(S_i, S_j)$ is defined to contribute more for the final similarity factor.

In this case, for each pair of candidate matching, a similarity notion is associated with it. Thus a set of corresponding planes can be denoted by $\{P_{t,i} \leftrightarrow$

$P_{t-1,j}, I_s(i,j)\}$. The uniqueness problem is solved by sorting all $\{P_{t,i} \leftrightarrow P_{t-1,j}, I_s(i,j)\}$ in increasing order of I_s then the pairing with the highest similar factor is automatically chosen and other candidate correspondences are rejected.

According to the experiments, we have found the above four tests to be much more effective to choose correct correspondences in consecutive frames, and most of the wrong matches are discarded. However, due to the existence of noise in point clouds, and large odometry errors, wrong matches are inevitable, which will cause a big divergence in the map.

To discard the wrong matches, further steps are necessary. In [22], the well-know RANSAC approach, which is famous for coping with noisy data and outliers, is used to discard the wrong matches. Its principle works as RANSAC, i.e., randomly select three matched feature pairs, which is the minimal number to compute a rigid transformation. According to the determined transformation, the pairs, which the pairwise Eucliden distance do not match, are considered as outliers and rejected, while other pairs are considered as inliers. Then the number of inliers is counted. These steps are executed iteratively and the transformation with most inliers is kept. However, RANSAC algorithm is suitable for the cases with a large number of feature pairs. In our work, in two successive frames, most of the times there are few pairs of corresponding planes found, therefore RANSAC could not be used here.

In order to further ward off the possibility of wrong matches, an plane matching consistency test is presented based on the assumption that the motion of robot between two measurement samples is rigid. Therefore, in principle the relative rotations estimated by different pairs of matching planes should be same.

Plane Matching Consistency Test

After the above four tests, the plane features in current frame $\mathcal{F}_t$ are divided into two categories: (1) successfully paired features. (2) planes that could not be matched to any plane feature in previous frame $\mathcal{F}_{t-1}$. The latter ones are considered as previously invisible, and labeled as new features.

Assuming a list $\mathcal{L}$ of corresponding pairs, along with their similarity measures $\{P_{t,i} \leftrightarrow P_{t-1,j}, I_s(i,j)\}, i = 1 \ldots N_t, j = 1 \ldots N_{t-1}$ are obtained. Obviously, the size of list $N_\mathcal{L}$ is not larger than N_t and N_{t-1}, i.e., $N_\mathcal{L} \leq \min(N_t, N_{t-1})$.

Given a pair of corresponding planes $P_{t,i} \leftrightarrow P_{t-1,j}$, we are able to estimate the relative robot rotation θ_t^{t-1} between frames $\mathcal{F}_t$ and $\mathcal{F}_{t-1}$, since plane $P_{t,i}$ and plane $P_{t-1,j}$ represent a same physical plane. θ_t^{t-1} is computed as:

$$\theta_{i,j} = \theta_i^p - \theta_j^p \tag{4.14}$$

Note that the time indices t and $t-1$ have been omitted for clarity. Thus a list

$\mathcal{L}_\theta$ of relative rotations $\{\theta_{i,j}\}$ will be obtained. Its size equals to the size of $\mathcal{L}$. In a rigid motion, the relative rotation angles should be same, i.e., their differences are expected to be close to 0. Thus their differences are considered to check the consistency between determined correspondences in $\mathcal{L}$.

For every two relative rotations in $\mathcal{L}_\theta$, their absolute difference $\delta\theta$ is computed. Since $N_\mathcal{L}$ rotations are obtained, there are $\frac{N_\mathcal{L} \cdot N_\mathcal{L} - 1}{2}$ differences between them. Then we can get the maximum one. If the maximum difference is close to 0, or less than a fixed threshold $\delta\theta_t$, all the relative rotations are considered as valid, i.e., all the correspondences are correct. Otherwise wrong matches are assumed to exist. According to the similarity definition, a larger similarity means the associated corresponding pair is more reliable. Based on this, the worst match indicated by the minimum similarity index is discarded. The same step is executed repeatedly until the maximum difference between relative rotations is less than the pre-defined threshold $\delta\theta_t$. The remained correspondences construct the final correspondence set:

$$\Omega^\star \triangleq \{P_{t-1,i} \leftrightarrow P_{t,j}, I_s(i,j)\} \tag{4.15}$$

The set $\Omega^\star$ may still contain non-unique correspondences, i.e., some planes in frame $\mathcal{F}_{t-1}$ may be mapped to more than one plane in current frame $\mathcal{F}_t$. For instance, $P_{t-1,j}$ is matched with more than one other plane in $\mathcal{F}_t$. The uniqueness problem is solved by sorting their similarity measures I_s. The pair with maximum similarity index is retained while other pairs are rejected. The fixed corresponding planes are labeled with a common index.

4.4 Corner Matching

The arrangement information between single features, coded as relations among features or their configuration, is an important aspect in correspondence problem, which may greatly help toward a solution. That means try to find matches not only based on the observed features, but also according to their organization. This is especially useful when features lack distinctive properties. Geometric constraints root in the fact that features are not isolated landmarks rather they are related by the structure of the scene.

In fact, arrangement information links individual features together. Therefore, it provides a larger context than a single feature. On the other hand, this larger context may be introduced as a new higher level (in the sense of abstraction) feature which is then much more unique and distinctive comparing to its elements. For instance, joining three intersecting planes together to form an orthogonal corner, which is

discussed in this thesis. Or even non-geometrical cases such as clustering SIFT features into templates which are then classified into objects.

A single orthogonal corner constructed by three intersecting planes is enough to lock all degrees of freedom in space, since it is a point feature and encodes the orientations of its parent planes. Therefore, complete poses tracking or localization is possible using corner features. In [51], in order to estimate the translation, the authors resort back to point features, which is identical features, which is essentially the same as ICP.

On the assumption that the floor is flat, the robot poses are presented in 2D transformations in $SE(2)$. Thus it is possible to project the orthogonal corners into $\vec{x}_r\vec{y}_r$-plane in robot reference frame and use their projections to present them. In this case, the values along $\vec{z}_r$ of all the corners are set to 0, and a corner can be thought as constructed by two vertical planes, shown in Figure 4.4. In Figure 4.4, a corner $C_{k,i}$ is constructed by two intersecting vertical planes $P_{k,i}$ and $P_{k,j}$, where k is the index of robot reference frame $\mathcal{F}_k$, i is the index of extracted features in frame $\mathcal{F}_k$ and (x, y) presents its position in 2D. Additionally, the orientation information of its parent planes is inherited to it. Thus the corner $C_{k,i}$ is presented as $C_{k,i}(x_i, y_i, \theta_x^p, \theta_y^p)$, where (x_i, y_i) defines its position, while (θ_x^p, θ_y^p) encodes its orientation information.

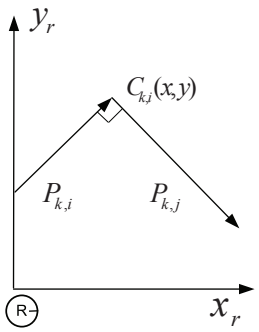


Figure 4.4: Representation of the projection of an orthogonal plane in 2D space

As for the problem of plane matching, corner matching consists in finding which corners represent the same physical corner and labelling them with a common index. Generally, as mentioned before, higher features as the orthogonal corners here, are used to initiate the search for correspondences and the result of their bindings are inherited to lower level features, i.e., corners are expected to be matched first, then their parent planes are associated. However, in our work, the matching process starts from plane matching problem which are the basis features, and corners correspondences are based on the plane matching results. The motivation for this choice

is twofold: (1) in the experiments of this thesis, less corners are observed than what is expected. For instance, in a long corridor environment, only four corners can be detected on the end sides of the corridor, while more planar patches are observed. (2) the plane matching is robust, most of the false correspondences are discard by using different tests presented before and all the determined correspondences are accepted as true. Actually, in practise, false correspondences are unavoidable.

Since all detected corners are made by intersecting planes, corner matching can be built on the basis of plane correspondences results. That means that corners which are constructed by the same planes are considered to be the same physical corner, i.e., given two corners $C_{t,i}$ and $C_{t-1,j}$, which constructed by plane-sets $\{P_{t,i1}, P_{t,i2}\}$ and $\{P_{t-1,j1}, P_{t-1,j2}\}$ respectively, if $P_{t,i1} \leftrightarrow P_{t-1,j1}$ and $P_{t,i2} \leftrightarrow P_{t-1,j2}$, then we have $C_{t,i} \leftrightarrow C_{t-1,j}$.

4.5 Relative Transformation Estimation

After plane matching and corner matching procedures, the corresponding planes and corners between frames $\mathcal{F}_k$ and $\mathcal{F}_t$ are determined. Given a pair of matching planes $P_{t,i}$ in frame $\mathcal{F}_t$ and $P_{t-1,j}$ in frame $\mathcal{F}_{t-1}$, denoted by a common index, the relative rotation θ_r between frames $\mathcal{F}_t$ and $\mathcal{F}_{t-1}$ is computed as:

$$\theta_r = \theta_j^p - \theta_i^p \tag{4.16}$$

where θ_j^p and θ_i^p are orientations of plane $P_{t-1,j}$ and $P_{t,i}$ respectively.

For a pair of matching corners $C_{t,i}$ and $C_{t-1,j}$, not only the relative rotation $\boldsymbol{R}_t^{t-1}$ but also the translation $\boldsymbol{t}_t^{t-1}$ can be estimated, since a corner fixes the position and orientation information at the same time. The two corners are related by

$$\begin{pmatrix} x_j \\ y_j \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\theta_r & -\sin\theta_r & 0 \\ \sin\theta_r & \cos\theta_r & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} + \begin{pmatrix} \Delta x \\ \Delta y \\ 0 \end{pmatrix} \tag{4.17}$$

where (x_j, y_j) and (x_i, y_i) are the position information of matching corners $C_{t-1,j}$ and $C_{t,i}$ respectively, $(\Delta x, \Delta y)$ are the relative translation $\boldsymbol{t}_t^{t-1}$, and θ_r is the relative rotation, as estimated by Eq.(4.16). Then the transformation matrix $\boldsymbol{t}_t^{t-1}$ is obtained as

$$\begin{pmatrix} \Delta x \\ \Delta y \\ 0 \end{pmatrix} = \begin{pmatrix} x_j \\ y_j \\ 1 \end{pmatrix} - \begin{pmatrix} \cos\theta_r & -\sin\theta_r & 0 \\ \sin\theta_r & \cos\theta_r & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} \tag{4.18}$$

According to Eq.(4.16) and Eq.(4.18), the pairwise transformations between robot poses can be computed and form the edges of a pose graph, that we have called the front-end part in SLAM.

4.6 Summary

In this Chapter, based on large planar surfaces and 3D orthogonal corners extracted from point clouds, a plane matching algorithm was presented for finding the plane correspondences, as well as orthogonal corners between consecutive frames.

On the assumption that the robot is operating in the plane, only vertical planar surfaces are considered in this Chapter. Thus it is possible to project them onto the $\vec{x}_r\vec{y}_r$-plane and use their projections, i.e., 2D lines to represent the planes. Different to the normal representations of line, we parameterized a line using two parameters (θ^p, d) to represent its relationship with the robot better, which are considered as the geometric information of planes.

The plane matching algorithm is performed by maximizing the similarity metric between a pair of planes within a search space to determine correspondences between planes. The search space is pruned using the followed criteria: odometric rotation agreement test, odometric translation agreement test, appearance similarity test, and size similarity test. To further discard wrong matches, a plane matching consistent test is given based on the fact that the estimated relative rotations by using different determined correspondences should be same, since the robot is rigidly moved between two poses. The determined plane correspondences are extended to the corner matching procedure. Based on the determined correspondences, the pose changes in orientation and position are estimated, forming edges between consecutive nodes in a pose graph. The formed pose graph is feed to a pose graph optimizer algorithm (SLAM back-end) to obtain a consistent and global trajectory of robot, which will be discussed in next Chapter.

Chapter 5

SLAM Back-end

Alignment between successive frames is a good method for tracking the robot trajectory over moderate distances. However, errors in alignment between particular pairs of frames, are unavoidable in practice. Moreover, noise in the obtained 3D point cloud, cause the estimation of robot poses to drift over time, leading to a divergence in the final map. This is more noticeable when the robot moves a long path. The drifting errors are accumulating as the robot moves, resulting a not globally consistent trajectory of robot. To create a globally consistent trajectory, a well assessed strategy is the so called pose graph optimization, referred as SLAM back-end. The objective of SLAM back-end is to estimate the robot's poses that maximize the likelihood of obtained constraints.

Global optimization is especially beneficial in case of a loop closure, i.e., when a robot drives in a loop and goes back to its starting location, since the loop closing edges in the graph allow to reduce the accumulated error. Loop closure is especially important in robotic mapping applications. It can be addressed as a place recognition problem. Without the ability to recognize the previously visited places, the position uncertainty of the robot increases without bound due to the continuous accumulation of dead-reckoning error and causes two representations of the same region in different locations. Place recognitions serve as constraints on the motion of the robot, allowing a correction of its dead-reckoning errors.

Loop closing is difficult for some reasons. For instance, the same location can look very different depending on which direction the exteroceptive sensor is pointing towards, or the environment changes caused by dynamic objects, such as moving humans or chairs. In recent years, loop closure detection has received considerable attention. Approaches to loop closure detection in 2D have broadly been presented, especially using 2D images [56] [18] [68] [73]. On the other hand, 3D point clouds have not been widely used for loop closure detection.

In this Chapter, loop closure detection is addressed using 3D point clouds and

the pose graph optimization are discussed. First a conceptual review about loop closure detection is presented to give a picture of existing solutions in general. Then we introduce a descriptor for 3D point cloud: viewpoint feature histogram (VFH) as described in [78] briefly, which will be used in our loop closure detection algorithm. Then the presented loop closure detection algorithm is explained thoroughly. After a brief discussion to the SLAM back-end problem, a linear pose graph optimization used in this work is introduced . Conclusions are presented at the end of this Chapter.

5.1 Loop Closure Detection

Loop closure detection can be seen as a place recognition problm. It consists of recognizing that the robot has returned to a previously visited location, i.e., determining that whether or not the current point cloud is similar to a previous one. Loop closure detection allows to refine the estimated map and robot trajectory, since the point clouds from a same location must be aligned with each other.

When the robot arrives at a previously visited location, i.e., forms a loop, the current point cloud $\mathcal{P}_t$, should resemble a point cloud $\mathcal{P}_k$ acquired previously, i.e., $t - k > 1$. A comparison is performed between point clouds $\mathcal{P}_t$ and $\mathcal{P}_k$ in order to determine whether or not a loop closure has occurred.

In this thesis, a loop closure detection based on a novel descriptor for 3D point cloud, named viewpoint feature histogram (VFH) and color histogram is presented. It is inspired by the strong recognition ability of VFH, while the usage of color histogram feature is to test the recognition further so to make loop closure detection more reliable.

VFH is a novel descriptor for representing a surface patch by a statistical histogram describing its geometry and viewpoint information. In contrast to 3D point clouds, it reduces the dimension of data and requires less memory. Therefore, working on features is easier than working with full point clouds and it is not computationally expensive.

5.1.1 Related Work

A large part of the related loop detection literature is focused on data from camera images and range data, in both 2D and 3D.

Laser sensors are widely used in SLAM. For example, in [38] [8], raw laser scans are used for relative pose estimation. Recently, a 2D loop closure detection algorithm introduced in [32] shows a good performance. It uses AdaBoost to create

a strong classifier composed from 20 weak classifiers, each of which describes a global feature of a 2D laser scan. The two most important weak classifiers are reported to be the area enclosed by the complete 2D scan and the area when the scan points with maximum range have been removed.

In [75], laser range scans are fused with images to form descriptors of the objects used as landmarks. The laser scans are used to detect regions of interest in the images through polynomial fitting of laser scan segments, while the landmarks are represented using visual features. Another example of loop closure detection algorithm, using both visual cues and laser data, is presented in [44]. Shape descriptors such as angle histograms and entropy are used to describe and match the laser scans. A loop closure is only accepted if both visual and spatial comparisons meet a match metric.

Work on vision-based loop closure detection have been presented in [17] [18]. A bag-of-words approach is presented, where scenes are represented as a collection of "visual words" (local visual features) drawn from a "dictionary" of available features. The appearance descriptor is a binary vector indicating the presence or absence of all words in the dictionary and it is used within a probabilistic framework together with a generative model of the observations. Another vision based loop closure detection approach is introduced in [10], where SURF features are extracted from images and classified as words using Tree-of-Words. A spatial constraint is imposed by checking nearest neighbors for each word in the images. In contrast to offline as in [10], a similar approach using visual words which is built online, for monocular SLAM is presented in [21].

Recently, methods for loop closure detection for 3D point clouds are introduced, which are similar to our case. In [65], a method based on the Normal Distribution Transform (NDT) [6] is presented. The NDT acts as a local descriptor of the point cloud. After discretizing space into bins, or cubes, the points in each bin are described as either linear, planar or spherical by comparing the size of the covariance matrix eigenvalues. Invariance to rotation is achieved after scans have been aligned according to the dominant planar surface orientation. Another method for loop closure detection from 3D range data is presented in [85]. The point cloud is transformed into a range image, from which features are extracted by computing the second derivative of the depth values in the range image. The Euclidean distance is used to compare the quality of match between pairwise features. Candidate transformations are calculated by matching features, and a score is assigned to evaluate how well the two scans are aligned. Rotation invariance is achieved by orienting image patches along the world $\vec{z}$ axis. According to the authors this does not restrict the performance of the method as long as the robot moves on a flat surface.

5.1.2 View Point Feature Histogram (VFH)

Surface Normals

Given a 3D point $\boldsymbol{p}_q$, a local feature representation that captures the geometry of the underlying sampled around $\boldsymbol{p}_q$ can be estimated by using its neighboring points $\mathcal{P}^k$. Surface normals, that describe its orientation in a coordinate system, are important properties of a surface. They are heavily used in many area such as computer graphics to determine a surface's orientation toward a light source for flat shading and other visual effects.

Many different normal estimation methods have been presented, and a comparison is presented in [50]. The simplest method is based on the first order plane fitting as proposed by [3]. The normal to a point on the surface is approximately determined by the normal of a plane tangent to the surface. In turn it becomes a least-square plane fitting estimation based on its neighboring points set $\mathcal{P}^k$.

We assume that the tangent plane is presented as a point $\bar{\boldsymbol{p}}$ and a normal vector $\vec{n}$, and the distance from a point $\boldsymbol{p}_i \in \mathcal{P}^k$ to the plane is defined as $d_i = (\boldsymbol{p}_i - \bar{\boldsymbol{p}}) \cdot \vec{n}$. The value of $\bar{\boldsymbol{p}}$ and $\vec{n}$ are computed in a least-square sense so that $d_i = 0$. The point $\bar{\boldsymbol{p}}$ is computed as the centroid of $\boldsymbol{p}_i \in \mathcal{P}^k$, shown as:

$$\bar{\boldsymbol{p}} = \frac{1}{k} \sum_{i=1}^{k} \boldsymbol{p}_i \tag{5.1}$$

where k is the number of point neighbors in $\mathcal{P}^k$. The normal $\vec{n}$ is estimated by analyzing the eigenvalues and eigenvectors of the covariance matrix $\boldsymbol{C} \in \mathbb{R}^{3\times3}$ of $\mathcal{P}^k$, expressed as:

$$\boldsymbol{C} = \frac{1}{k} \sum_{i=1}^{k} (\boldsymbol{p}_i - \bar{\boldsymbol{p}})(\boldsymbol{p}_i - \bar{\boldsymbol{p}})^T, \quad \boldsymbol{C}\vec{\mathrm{v}}_j = \lambda_j \vec{\mathrm{v}}_j, \qquad j \in \{0, 1, 2\} \tag{5.2}$$

λ_j is the j-th eigenvalue of the covariance matrix, and $\vec{\mathrm{v}}_j$ the corresponding j-th eigenvector.

$\boldsymbol{C}$ is symmetric and positive semi-definite, and its eigenvalues are real numbers $\lambda_j \in \mathbb{R}$. The eigenvectors $\vec{\mathrm{v}}_j$ form an orthogonal frame, corresponding to the principal components of P^k. If $0 \leq \lambda_0 \leq \lambda_1 \leq \lambda_2$, the eigenvector $\vec{\mathrm{v}}_0$ corresponding to the smallest eigenvalue λ_0 is therefore the approximation of $+\vec{n} = \{n_x, n_y, n_z\}$ or $-\vec{n}$. According to the above description, the estimated normal is dependent on the size of neighborhood P^k. So the choice of k is important in order to suitably estimate the normal.

Figure 5.1 presents an example of surface normal estimation for points lying on a small box. As shown, the resultant surface normals can suitably describe the geometric feature of the surface surrounding the detected points.

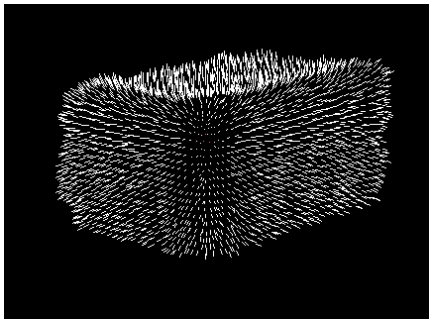


Figure 5.1: An example of surface normals estimation for points lying on a 3D box. The white arrows show the direction of estimated normals.

Viewpoint Feature Histogram

Viewpoint feature histogram (VFH) is an extension of fast point feature histogram (FPFH) [76]. It combines FPFH with viewpoint component so to inherit the strong recognition ability of FPFH, meanwhile encode the relationship between the viewpoint and surface normals on the query point cloud or object.

As its name implies, a point feature histogram representation (PFH) presented in [80], is a statistic histogram which encodes the relationships between every pair of points and their normals on a surface patch. Given a pair of 3D points $(\boldsymbol{p}_i, \boldsymbol{p}_j)$, their estimated surface normals are $\vec{n}_i$ and $\vec{n}_j$, respectively. The relationship between the normals is defined as the angular deviations $\{\alpha, \beta, \gamma\}$, which are estimated as:

$$\alpha = \vec{\mathrm{v}} \cdot \vec{n}_j \tag{5.3}$$

$$\beta = \vec{\mathrm{u}} \cdot \frac{\boldsymbol{p}_j - \boldsymbol{p}_i}{d} \tag{5.4}$$

$$\gamma = \arctan(\vec{\mathrm{w}} \cdot \vec{n}_j, \vec{\mathrm{u}} \cdot \vec{n}_j) \tag{5.5}$$

where $\vec{\mathrm{u}}, \vec{\mathrm{v}}, \vec{\mathrm{w}}$ represent a Darboux frame coordinate system chosen at $\boldsymbol{p}_i$, and d is the Euclidean distance between points $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$. Then the point feature histogram captures all the sets of α, β, γ between all pairs of $(\boldsymbol{p}_i, \boldsymbol{p}_j), i, j = 1, 2, \cdots, n$ on a surface patch and bins the results in a histogram. The bottom part of Figure 5.2 [78] presents the definition of the Darboux frame and a graphical representation of the three angular features between pairwise points.

If $\boldsymbol{p}_j$ is only defined as k-nearest neighbor points of $\boldsymbol{p}_i$ so that the computation time will be reduced, shown by subset of points in Figure 5.2 the obtained histogram will be a fast point feature histogram.

The viewpoint component is built by collecting a histogram of the angles that translating the central viewpoint direction to each of the normals on the patch surface. Similar to PFH, it measures the relative pan, tilt and yaw angles between the viewpoint direction at the central point and each of the normals.

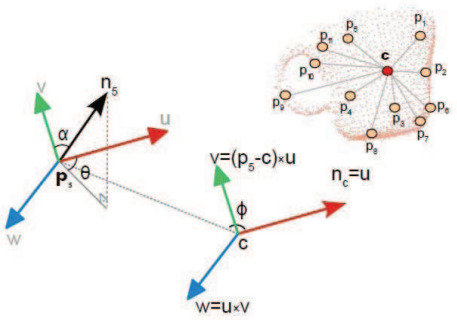


Figure 5.2: The extended fast point feature histogram collects the statistics of the relative angles between the surface normals at each point to the surface normal at the centroid of the object. The bottom left part of the figure describes the three angular feature for an example pair of points, while the top right part shows a surface patch which the points lying in.

Figure 5.3 shows an example of VFH presentation for a point cloud. Notice that in Figure 5.3 the VFH is divided into four sub-histograms, where the first sub-histogram presents the viewpoint component, while the other three correspond to the FPFH component.

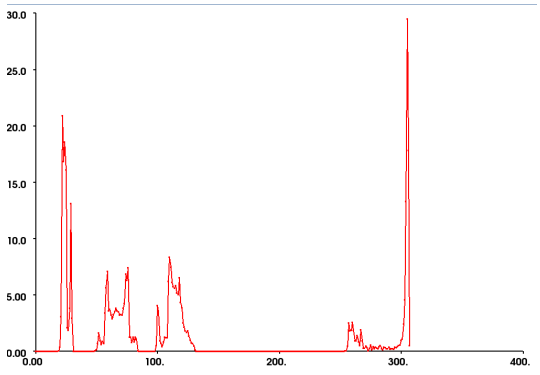


Figure 5.3: An example of VFH presentation of an obtained point cloud.

5.1.3 Loop Closure Detection using VFH

Our loop closure detection algorithm uses the same principle used in other approaches: detect the loop closure by comparing the pairwise point clouds. In order to reduce the computation time, we define keyframes, which are a subset of the overall aligned frames. Moreover, the use of keyframes can keep the graph relatively sparse.

In [42], the keyframes are defined based on visual overlaps. Given a frame, when it fails to match against the previous keyframes, it is determined as a new keyframe. In this work, the frames in which 3D orthogonal corners are detected are defined as keyframes, since the point clouds containing 3D orthogonal corners encodes more geometry information and are more distinguishable. Moreover, the rigid transformation between the determined loop closing frames, can be estimated by the identical corners in both frames, which is similar to the registration using corner matching discussed in Chapter 4. And the point clouds associated with them are defined as key point clouds.

Each time a keyframe is found, we attempt to compare it with each previous keyframe and detect whether or not a loop closure has occurred. A loop closure is determined if the detected frames meet the predefined geometrical consistent and color-appearance consistent conditions. Then the pose change between these two frames are estimated, and added to the graph.

Thus, the whole loop closure detection algorithm consists in the following two steps:

1. detecting whether or not a loop closure has occurred via comparing features of pairwise point clouds.

2. finding the corresponding corners which present an identical geometrical corner, which is used to estimate the rigid transformation between two frames.

The two steps are separately explained below.

Loop Closure Determination

A current frame $\mathcal{F}_t$, associated with point cloud $\mathcal{P}_t$, is labeled as a keyframe if one or more corners are detected in $\mathcal{P}_t$, Assuming all the previous keyframes are saved in set $\boldsymbol{F}_t = \{\mathcal{F}_k, 0 < k < t-1\}$, and their corresponding VFH features form the set $\boldsymbol{V}_t = \{V_k, 0 < k < t-1\}$, while their color-appearance histograms are presented as $\boldsymbol{H}_t = \{H_k, 0 < k < t-1\}$. Our approach detects loop closures by matching current frame against the previously collected frames. To facilitate the comparison of two frames $\mathcal{F}_t$ and $\mathcal{F}_k$, the both features are considered. The approach consists in the following two steps:

1. select the frames that meet the geometric consistent test by comparing their VFH features. The underlying idea here is that point clouds acquired at the same location will have similar VFH feature values V_t and V_k.

2. select the final frame that has the largest color appearance consistency. The underlying idea here is that point clouds acquired at the same location should have similar color information.

In the first step, the algorithm searches for a subset of corresponding VFH for the geometric persistent feature histogram of the query VFH. In order to quantify the different between two VFH features V_t and V_k, we compute the Chi-square distance d_V between them. The Chi-square distance is defined as:

$$d_V^2 = \frac{1}{2}\sum_i \frac{(V_{t,i} - V_{k,i})^2}{V_{t,i} + V_{k,i}} \tag{5.6}$$

where $V_{t,i}$ and $V_{k,i}$ present the i-th bin value of V_t and V_k respectively.

To select the frames which have similar VFH features with the current one efficiently, a kd-tree is created in the VFH feature histograms space, and for each query VFH V_t, a K-nearest neighbor search in the previous VFH set. The K frames, saved in set $\boldsymbol{F}_K$, with most similar VFH features are returned with sorted Chi-square distance in increasing order. To discard wrong matches, a pre-defined maximum threshold Δd_v is used further to choose potential candidate frames for $\mathcal{F}_t$, i.e., for every frame $\mathcal{F}_k \in \boldsymbol{F}_K$, if its associated Chi-square distance is less than Δd_v, it is added to the potential candidate set $\boldsymbol{F}_c$, expressed as:

$$\boldsymbol{F}_c = \{\mathcal{F}_k \mid d_v(V_k, V_t) \leq \Delta d_v, \mathcal{F}_k \in \boldsymbol{F}_K\} \tag{5.7}$$

In the second phase of the approach, the final resolved frame is selected by using color appearance feature from $\boldsymbol{F}_c$. Similar to the discussion in Chapter 4, here the correlation between color histograms in RGB space is used again to measure the similarity between the detected frames (or point clouds). If the point clouds are obtained at same location, their color appearances are similar, thus the correlation should be close to 1. The use of color histogram is not only to choose the final matching frame, but also make the loop closure detection more reliable. In this case, the frame in $\boldsymbol{F}_c$ with the largest color covariance, greater than a fixed minimum threshold ΔH_c, is determined to be the final frame resembling to the query frame $\mathcal{F}_t$. It will be denoted as $\mathcal{F}_t \leftrightarrow \mathcal{F}_k$. While if no frame passes the two persistent tests, it means that the robot has moved to a new location. Actually, if the size of $\boldsymbol{F}_c$ is 0, the second step is not needed.

Corner Matching

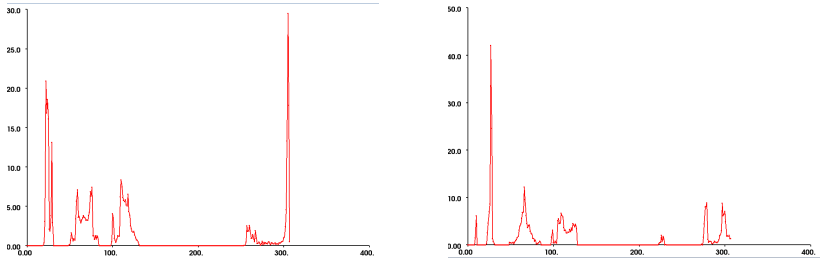
Defining the frames, in which 3D orthogonal corners are detected as keyframes, has the advantage that we can make use of the included corners to estimate the rigid

roto-translation between the determined loop closures, since corners can lock all the degrees in $SE(2)$. Thus we do not need to resort back to other registration approaches, like point-to-point ICP.

We consider a pair of frames presenting a same scene, $\mathcal{F}_t$ from which the indexed corner-set C_t is observed, and $\mathcal{F}_k$ from which the indexed corner-set C_k is observed. In order to keep the notation consistent, we assume $k < t$. Here the goal is to estimate the relative transformation $\boldsymbol{R}_t^k, \boldsymbol{t}_t^k$, resolved in local reference frame $\mathcal{F}_k$, by using corner-sets C_t and C_k.

To estimate the transformation by using corners, first we have to find the correspondences, presenting the same physical corner, between two corner-sets. A matching algorithm has to be applied to find the correct correspondences between corner-sets C_t and C_k. This is similar to the corner matching problem discussed in Chapter 4, but it is more challenging. In Chapter 4, the odometry information is used to find the potential candidates and discard wrong matches. However, since with the typical odometry errors the pose estimate will be totally wrong after a long distance, the odometry information can not be used here.

The corners in a same frame look different depending on their relative relationship with respect to the direction the scanner is pointing towards. As presented before, VFH feature encodes the geometrical structure and viewpoint information in the meanwhile. Therefore, for two corners in a same frame, in principle, their corresponding VFHs will be different. Figure 5.4(a) and Figure 5.4(b) show the VFH features of two corners in a same frame, respectively. It can be seen that Figure 5.4(a) is quite different from Figure 5.4(b), satisfying the expected. Based on this, here we reuse the VFH feature again to find the similar corners in the detected two loop closing frames.



(a) VFH of corner $C_{t,i}$ (b) VFH of corner $C_{t,j}$

Figure 5.4: Examples of VFH representations for two corners detected in the frame $\mathcal{F}_t$.

Similar to above, the corners with the most similar VFH features, i.e., the lowest Chi-square distance, are determined as corresponding corners. In our case, gener-

ally, there is only one or two corners detected in a frame, thus VFH feature is enough to determine the correct corner correspondences and color feature is not used here. The experimental results validates the presented approach, and VFH feature can find the correct corner correspondences. The experimental results will be reported in Chapter 6.

5.2 SLAM Back-end

In graph-based SLAM, the poses of the robot are modeled by nodes in a graph and labeled with their position in the environment. Spatial constraints between poses that estimated from scan-matching or provided from odometry measurements are encoded in the edges between the nodes. Each node in the graph represents a robot position and a measurement acquired at that position. In Chapter 4, a plane matching approach was presented to construct spatial constraints between consecutive poses from sensor data. While in the above section in this Chapter, a loop closure detection algorithm combining VFH feature and color histogram feature, was introduced, which allows the robot has the ability to recognize previously-visited places and estimate the transformations between the two poses. They mainly focus on extracting the constraints from sensor data and is often referred to as the SLAM front-end. In contrast to that, the SLAM back-end aims at correcting a pose graph given all constraints.

The goal of SLAM back-end is to find the best poses configuration given the constraints. A number of optimization algorithms based on SLAM back-ends are readily available as open source libraries. For instance, g^{2}o [56], TORO [37], MTK [91]. Choosing a suitable optimizer is important to obtain a consistent and accurate trajectory of the robot. Targeting to our work, a linear approximation for the pose graph configuration proposed in [13] [12] is selected to optimize the built pose graph.

5.2.1 Problem Statement

Given the built pose-graph from SLAM front-end, the objective of SLAM back-end is to find the configuration of the robot poses that best satisfies the constraints. Let us call the robot poses $\mathbf{x} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, where $\mathbf{x}_i$ describes the pose of node i. $\mathbf{x}_i$ is in the form $\mathbf{x}_i = [{p_i}^T \ \theta_i]^T \in SE(2)$, where $p_i \in \mathbb{R}^2$ is the Cartesian position of the i-th pose, and θ_i is its orientation.

The virtual relative pose between the node i and node j is assumed as $\tilde{\xi}_{ij}$. Note that $\tilde{\xi}_{ij}$ is expressed in local reference frame of $\mathcal{F}_i$, and it makes the observation

acquired from i maximally overlap with the observation acquired from j. However, the relative pose measurement between the two nodes are affected by noise, i.e., $\xi_{ij} = \tilde{\xi}_{ij} + \epsilon$, where $\epsilon_{ij} \in \mathbb{R}^3$ is a zero mean Gaussian noise, i.e., $\epsilon_{ij} \sim \mathcal{N}(\mathbf{0}, \mathcal{C}_{ij})$, $\mathcal{C}_{ij}$ is 3 by 3 covariance matrix.

The pose graph built in front-end procedure is indicated as $\mathcal{G}(\mathbf{x}, \varepsilon)$, where ε is the graph edges, containing the unordered node pairs (i, j) such that a relative pose measurement exists between i and j. Once the relative pose measurements and the corresponding uncertainty are given, the robot is required to estimate its pose configuration $\mathbf{x}$ in a given global reference frame. Usually the initial pose of the robot is set to be the origin of the global reference frame, i.e., $\mathbf{x}_0 = [0, 0, 0]^\top$.

The goal of SLAM back-end is to determine the configuration of the robot poses $\mathbf{x}^*$ that minimizes the negative likelihood of all the observation. Generally it is expressed as:

$$f(\mathbf{x}) = \sum_{(i,j)\in\varepsilon} \boldsymbol{e}(\mathbf{x}_i, \mathbf{x}_j, \xi_{ij})^\top \Omega_{ij} \boldsymbol{e}(\mathbf{x}_i, \mathbf{x}_j, \xi_{ij}) \tag{5.8}$$

where $\boldsymbol{e}(\mathbf{x}_i, \mathbf{x}_j, \xi_{ij})$ is a function that computes the difference between the expected observation $\tilde{\xi}_{ij}$ and the real observation ξ_{ij} gathered by the robot, Ω represents the information matrix of the virtual measurement $\tilde{\xi}_{ij}$ between poses $\mathbf{x}_i$ and $\mathbf{x}_j$. Since the measurement noise is assumed as Gaussian noise, the likelihood function (5.8) is equivalent to minimize the sum of the weighted residual errors:

$$f(\mathbf{x}) = \sum_{(i,j)\in\varepsilon} (\tilde{\xi}_{ij} - \xi_{ij})^\top \mathcal{C}_{ij}^{-1} (\tilde{\xi}_{ij} - \xi_{ij}) \tag{5.9}$$

Here $\Omega_{ij} = \mathcal{C}_{ij}^{-1}$. The full SLAM problem is hence formulated as a minimization of the nonlinear non-convex function (5.9), i.e., the optimal configuration is $\mathbf{x}^* = \arg\min f(\mathbf{x})$ [12].

5.2.2 A Linear Pose-Graph Optimizer

A linear pose graph optimizing algorithm has been recently presented in [12]. The work is extended in [13] relaxing the hypothesis that measurement covariance matrices have a block diagonal structure. Under the assumption that the relative position and the relative orientation are independent, the full SLAM problem is approximated to be a closed-form solutions. The approaches need no initial guess for optimization and can be solved in a single step instead of iteratively. Its general idea is to separate the estimation of orientation and position. By estimating both quantities separately, the optimizing problem is divided into two linear problems.

Each relative pose measurement consists ξ_{ij} two components: relative position and relative orientation. Thus each measurement ξ_{ij} is rewritten as $\xi_{ij} = [\Delta_{ij}^l, \delta_{ij}]^\top$,

where Δ_{ij}^l corresponds to the relative position, while δ_{ij} presents the relative orientation where the superscript l denotes that the relative position vector is expressed in a local frame. The relative rotation measurement δ_{ij} is regularized by adding a suitable multiple of 2π, i.e., $\{\delta_{ij}\} = \delta_{ij} + 2k_{ij}\pi$, where k_{ij} is called regularization term. Thus the cost function (5.9) can be rewritten as

$$f(\mathbf{x}) = \sum_{(i,j)\in\varepsilon} \begin{bmatrix} R_i^\top(\rho_j - \rho_i) - \Delta_{ij}^l \\ (\theta_j - \theta_i) - \delta_{ij} \end{bmatrix}^\top \mathcal{C}_{ij}^{-1} \begin{bmatrix} R_i^\top(\rho_j - \rho_i) - \Delta_{ij}^l \\ (\theta_j - \theta_i) - \delta_{ij} \end{bmatrix} \tag{5.10}$$

where $R_i \in \mathbb{R}^2$ is a planar rotation matrix of angle θ_i. The relative position information and the relative orientation measurements are assumed independent, i.e. $\mathcal{C}_{ij} = \text{diag}(\mathcal{C}_{\Delta_{ij}^l}, \mathcal{C}_{\delta_{ij}})$. Under this assumption the cost function $f(\mathbf{x})$ becomes:

$$\begin{aligned} f(\mathbf{x}) = &\sum_{(i,j)\in\varepsilon} \left[R_i^\top(\rho_j - \rho_i) - \Delta_{ij}^l\right]^\top \mathcal{C}_{\Delta_{ij}^l}^{-1} \left[R_i^\top(\rho_j - \rho_i) - \Delta_{ij}^l\right] \\ &+ \sum_{(i,j)\in\varepsilon} \left[(\theta_j - \theta_i) - \delta_{ij}\right]^\top \mathcal{C}_{\delta_{ij}}^{-1} \left[(\theta_j - \theta_i) - \delta_{ij}\right] \end{aligned} \tag{5.11}$$

To put the previous formulation in a more compact form, the relative position measurements are stacked in the vector $\Delta = \left[(\Delta_1^l)^\top, (\Delta_2^l)^\top \ldots, (\Delta_m^l)^\top\right]^\top$, while all the relative orientation measurements are in the vector $\delta = [\delta_1, \delta_2, \ldots, \delta_m]^\top$. Accordingly, the information matrix $\Omega_{ij}, (i,j) \in \varepsilon$ is reorganized into a large matrix. Then the cost function (5.11) can be written as:

$$\begin{aligned} f(\mathbf{x}) = &(A_2^\top \rho - R\Delta^l)^\top (R\mathcal{C}_{\Delta^l} R^\top)^{-1} (A_2^\top \rho - R\Delta^l) + \\ &(A^\top\theta - \delta)\mathcal{C}_\delta^{-1}(A^\top\theta - \delta) \end{aligned} \tag{5.12}$$

where:

- A is the reduced incidence matrix of graph $\mathcal{G}$;
- $A_2 = A \otimes \mathbf{I}_2$ is an expanded version of A;
- $R = R(\theta) \in \mathbb{R}^{2m,2m}$ is a block diagonal matrix, whose nonzero entries are in positions $(2k-1, 2k-1), (2k-1, 2k), (2k, 2k-1), (2k, 2k),\ k = 1, \ldots, m$, such that, if the k-th measurement correspond to the relative pose between i and j, then the k-th diagonal block of R is a planar rotation matrix of an angle θ_i.

The minimization of the cost function (5.12) is equivalent to find the solution satisfying the following two constraints:

$$\begin{cases} A_2^\top \rho = R(\theta)\Delta^l \\ A^\top\theta = \delta \end{cases} \tag{5.13}$$

In principle, the cost function (5.12) will be zero when a solution exactly satisfies the constraints presented in (5.13). Otherwise, a minimum residual errors is searched with the constraints (5.13).

Notice that the seconde constraints, including the relative orientation measurement, is linear in the unknown variable θ. An obtained result θ provides an estimate of the relative measurement for the first equation in (5.13). Thus the whole optimization procedure is divided into three phases:

1. consider the second constraint, solving the following linear estimation problem

$$A^\top \theta = \delta \tag{5.14}$$

from which the suboptimal orientation estimate $\hat{\theta}$ and its covariance matrix can be obtained. The Eq. (5.14) is a standard linear estimation problem. According to the linear estimation theory, the optimal $\hat{\theta}$ and the corresponding covariance are:

$$\hat{\theta} = (A\mathcal{C}_\Delta^{-1}A^\top)^{-1}A\mathcal{C}_\Delta^{-1}\theta \qquad \mathcal{C}_{\hat{\theta}} = A\mathcal{C}_\Delta^{-1}A^\top \tag{5.15}$$

respectively.

Therefore, using $\hat{\theta}$ as the actual nodes' orientation, an estimate for the position $\hat{\rho}$ is obtained, expressed as:

$$\hat{\rho} = \left[A_2(\hat{R}\mathcal{C}_{\Delta^l}\hat{R}^\top)^{-1}A_2^\top\right]A_2(\hat{R}\mathcal{C}_{\Delta^l}\hat{R}^\top)^{-1}\hat{R}\Delta^l \tag{5.16}$$

where $\hat{R} = R(\hat{\theta})$. It is important to note that the first equation in (5.14) also constraints the orientations of the robot, thus the estimate $\hat{\mathbf{x}} = [\hat{\rho}^\top \;\; \hat{\theta}^\top]^\top$ is a suboptimal solution and needs to be corrected later.

2. estimate the relative position measurements in the global reference frame:

$$z = \begin{bmatrix} \hat{R} & \mathbf{0}_{2m\times n} \\ \mathbf{0}_{2m\times n}^\top & I_n \end{bmatrix}\begin{bmatrix} \Delta^l \\ \hat{\theta} \end{bmatrix} = \begin{bmatrix} g_1(\Delta^l,\theta) \\ g_2(\theta) \end{bmatrix}_{\theta=\hat{\theta}} \tag{5.17}$$

compute the corresponding uncertainty:

$$\mathcal{C}_z = H\begin{bmatrix} \mathcal{C}_{\Delta^l} & \mathbf{0}_{2m\times n} \\ \mathbf{0}_{2m\times n}^\top & \mathcal{C}_{\hat{\theta}} \end{bmatrix}\begin{bmatrix} \rho \\ \theta \end{bmatrix}H^\top \tag{5.18}$$

where H is the Jacobian of the transformation in (5.17):

$$H = \begin{bmatrix} \frac{\partial g_1}{\partial \Delta^l} & \frac{\partial g_1}{\partial \theta} \\ \frac{\partial g_2}{\partial \Delta^l} & \frac{\partial g_2}{\partial \theta} \end{bmatrix} = \begin{bmatrix} \hat{R} & J \\ \mathbf{0}_{2m\times n} & I_n \end{bmatrix} \tag{5.19}$$

In this phase, z is the estimate of the relative position measurements in the global reference frame, and it is formed as: $z = [(\Delta^g)^\top \quad \hat{\theta}^\top]^\top$ where $\Delta^g = \hat{R}\delta^l$ is the vector containing the relative node position expressed in the absolute reference frame $\mathcal{F}_0$.

3. As mentioned before, the estimated $\hat{\mathbf{x}}$ constitutes a suboptimal solution. Thus in the last phase, it is corrected, leading to get the minimum of the cost function, i.e.,

$$\theta^* = \hat{\theta} + \tilde{\theta}, \qquad \rho^* = \hat{\rho} + \tilde{\rho} \tag{5.20}$$

in which $\tilde{\theta}$ and $\tilde{\theta}$ are first-order correction terms.

Given z in (5.17) and $\mathcal{C}_z$ in (5.18), it is able to solve the linear estimation problem in the unknown $\mathbf{x} = [\rho^\top \quad \theta^\top]^\top$, shown as:

$$z = \begin{bmatrix} A_2^\top & \mathbf{0}_{2m\times n} \\ \mathbf{0}_{2m\times n}^\top & I_n \end{bmatrix} \begin{bmatrix} \rho \\ \theta \end{bmatrix} = B^\top \mathbf{x} \tag{5.21}$$

from which the solution $\mathbf{x} = [(\rho^*)^\top \quad (\theta^*)^\top]^\top$ and the corresponding uncertainty can be retrieved. The optimized poses is obtained as:

$$\mathbf{x}^* = \begin{bmatrix} \rho^* \\ \theta^* \end{bmatrix} = (B\mathcal{C}_z^{-1}B^\top)^{-1}B\mathcal{C}_z^{-1}z \tag{5.22}$$

5.3 Summary

This Chapter presented the SLAM back-end algorithms, which is used to find a configuration of the robot's poses that is maximally consistent with the measurement.

Loop closing is a form of place recognition that is central to the task of map building: it prevents the unbounded growth of dead-reckoning error. In this Chapter, we described a loop closure detection algorithm from 3D point clouds by comparing VFH descriptors and color histograms. Compared to 3D point clouds, VFH descriptors compress the input point clouds' geometry information into meaningful statistic histograms while keeping the viewpoint component, thus it reduces the dimension and store space of the data.

For avoiding expensive computation cost, the frames in which orthogonal corners are detected are defined as keyframes, since orthogonal corners fix the position and orientation at the same time. Similar to many other approaches, the problem is solved via comparing the features of pairwise views. Each time when a keyframe is detected, we attempt to detect a loop closure with each previous keyframes. A closure is detected if enough geometrically and color appearance consistent between

pairwise frames matching. If so, VFH feature is used again to find the corresponding corners between the two frames, and the estimated roto-translation is added to the graph representing this newly discovered constraint.

Most of the optimization require the availability of an initial guess for nonlinear optimization. In order to get a global solution, a sufficiently accurate initial guess is needed. In this work, instead, a linear approximation for the pose graph optimization has been applied that does not require any initial guess, and was shown to be accurate in practise. In this Chapter, its theoretical background was briefly introduced.

The next Chapter presents the real 3D mapping results.

Chapter 6

Experiments

Some exemplary experiments results are presented in this Chapter in order to evaluate the performance of the proposed plane-based 3D mapping algorithm. The experiments have been carried out with a Pioneer P3DX wheeled robot, shown in Figure 6.1 , inside Dipartimento di Automatica e Informatica at Politecnico di Torino. The robot is only equipped with wheel encoders, a laser range finder (SICK LMS-200), and a Microsoft Kinect sensor, where the laser range finder is only used for obstacle avoidance, while the wheel encoders and the Microsoft Kinect sensor are used for this work. The wheel encoders can provide initial odometry information about the robot poses, and the Microsoft Kinect sensor is used to collect 3D point clouds of the environment. Notice that the Microsoft Kinect camera is mounted on the top of the robot, parallel to the ground floor, at an angle of 45° around the vertical. This allows the Microsoft Kinect to sense the surrounding walls, doors etc., better.

During the different experiments, instead of moving in a stop-and-go manner to collect the data when the robot stands still, the robot moves continuously, since the Microsoft Kinect camera is able to provide both color images and dense depth maps at full video frame rate. When the robot passes through a door, or enter into a new space, the robot is manually guided through with slow speed such that observation samples remain proper.

Since the test environment is flat everywhere with the exception of some small ramps, it is believed that it is suitable to represent the robot's poses in $SE(2)$, i.e., $\mathbf{x}_i = [x, y, \theta]$, where $\mathbf{x}_i$ is the robots pose at time i. The map is presented in 3D constructed by attaching each acquired point cloud to its corresponding estimated robot pose. In the sections below, after a simple test for the plane matching registration, three 3D mapping experimental results are given, and the performance of the plane-based mapping algorithm is analyzed in a qualitative way by comparing their reconstructed 3D map with the real scenarios, respectively.

Figure 6.1: The robot used for the experiments is a differential-drive mobile robot equipped with a multitude of sensors: wheel encoders, a Sick LMS-200 laser range scanners and a Microsoft Kinect camera. As mentioned earlier in the text, only the wheel encoders and the Microsoft Kinect camera are used for this work. The laser scanner is only used for obstacle avoidance. For each frame, the Microsoft Kinect camera obtains 307,200 (640×480) 3D points, corresponding to the dimension of the acquired image.

In this thesis, Point Cloud Library (PCL) [79] is used for point cloud processing, while OpenCV Library [9] is used for histogram processing. During the experiment, in order to make the plane matching procedure more reliable, planes with an area smaller than 0.25 m^2 and with a number of supporting points smaller than 500 are filtered out.

6.1 Plane Matching Registration Experiments

As mentioned before, to robustly build correspondences between plane-sets in two consecutive frames, is a difficult task. In Chapter 4, a plane matching algorithm was presented for finding the plane correspondences, as well as orthogonal corners between consecutive frames. Based on the determined correspondences, the pose changes in orientation and position are estimated, forming edges between consecutive nodes in a pose graph.

Here in order to estimate the performance of the plane matching approach, registration tests between ten sets of successfully paired consecutive frames were preformed. For each set of paired frames, the relative roto-translation information are computed based on the correspondences between the features detected in these two frames. Therefore we were able to test the plane matching algorithm by registering pairwise frames together and evaluating the registration results. Since the relative roto-translation between two frames is unknown, it is difficult to evaluate the registration in quantitative way. Here the results of the pairwise registrations are manually inspected, as in [71].

Meanwhile, a comparison with two baseline registration algorithms was preformed. The standard ICP algorithm was used as one of the reference implementations, while the SAmple Consensus Initial Alignment (SAC-IA) with FPFH provided a second point of comparison. In Chapter 2, their basic theories were introduced briefly.

The performance of the discussed registration algorithms over all 10 test pairwise frames is summarized in Figure 6.2 and Table 6.1. Notice that no corners are included in Figure 6.2, in the first five frame sets. Therefore, for these frame sets, only the relative orientation information could be estimated. Their registration results are evaluated by checking whether or not the detected corresponding surfaces are parallel. If the corresponding surfaces are parallel, the registration is considered as successful. Otherwise it is labeled as failed. While for the last five frame sets, corresponding corners are detected. Thus, both relative orientation and position information could be estimated. A registration is judged to have failed only if there are severe displacements between the two supposedly registered frames.

From Figure 6.2, it can be seen that for all the registration results using plane matching registration method, there are not obvious misalignment. And all the registration results are labeled as successful in Table 6.1.

The performance of the base-line ICP algorithm is very good on scan pairs sets $\{\mathcal{F}_{21}, \mathcal{F}_{22}\}$, $\{\mathcal{F}_{10}, \mathcal{F}_{11}\}$, $\{\mathcal{F}_{82}, \mathcal{F}_{83}\}$ and $\{\mathcal{F}_{200}, \mathcal{F}_{201}\}$, but extremely poor on frames $\{\mathcal{F}_{143}, \mathcal{F}_{144}\}$, even worse than the odometry. While for the pairwise frame sets $\{\mathcal{F}_1, \mathcal{F}_2\}$, $\{\mathcal{F}_{105}, \mathcal{F}_{106}\}$ and $\{\mathcal{F}_{141}, \mathcal{F}_{142}\}$, there is no obvious improvement with respect to the odometry ones. This occurs when moderate or big pose changes happening between two successive views. Two factors are subject to the failure results: (1) large overlap but few features in the frames, which represents a typical failure case for ICP; (2) bad initial guesses from odometry information. ICP is known to perform better if a good initial guess is given, especially for the rotation. When a bad pose is given as the initial guess for the iteration, generally, the outcome is not improved. On the contrary, it worsens the outcomes. The registration between frames $\{\mathcal{F}_{143}, \mathcal{F}_{144}\}$ is an example of this behavior.

The second reference approach SAC-IA with FPFH features did not deliver promising results for all the registration results. With respect to ICP, the big error in odometry in frame sets $\{\mathcal{F}_{143}, \mathcal{F}_{144}\}$ is modified, and corresponding corners are overlapped totally. Also for the frames set $\{\mathcal{F}_{10}, \mathcal{F}_{11}\}$, the registration results improved significantly. However, there are still small misalignments in the registration of sets $\{\mathcal{F}_1, \mathcal{F}_2\}$ and $\{\mathcal{F}_{105}, \mathcal{F}_{106}\}$. Especially for the registration of frames $\{\mathcal{F}_{249}, \mathcal{F}_{250}\}$, SACIA performs worst. Note that according to the theory of SACIA, it performs better when the scans have more features. While for pairwise scans $\{\mathcal{F}_{249}, \mathcal{F}_{250}\}$, only one plane points are in one scan, and their geometry features are similar. It is difficult to find a good transformation between estimated features.

Compared with ICP and SAC-IA, results showed that the proposed plan matching algorithm performs better over all with no gross misalignment for all the test sets.

6.2 3D Mapping Experiments

In this section, three experiments are reported to evaluate the 3D mapping using plane-based SLAM algorithm, presented in this thesis.

6.2.1 Scenario 1: A Long Corridor

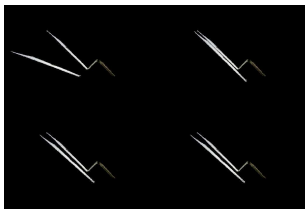
The DAUIN laboratory floor is used as the testing zone. Actually it consists of a long, narrow corridor with office rooms on both sides. The corridor is divided into

(a) pair $\mathcal{F}_1$ and $\mathcal{F}_2$

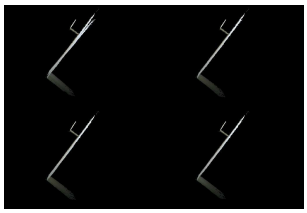
(b) pair $\mathcal{F}_{105}$ and $\mathcal{F}_{106}$

(c) pair $\mathcal{F}_{21}$ and $\mathcal{F}_{22}$

(d) pair $\mathcal{F}_{141}$ and $\mathcal{F}_{142}$

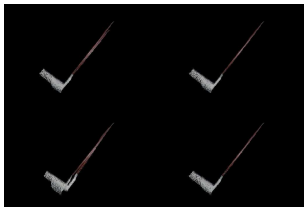


(e) pair $\mathcal{F}_{249}$ and $\mathcal{F}_{250}$

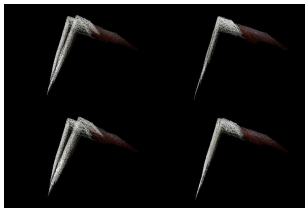


(f) pair $\mathcal{F}_0$ and $\mathcal{F}_1$ and 1

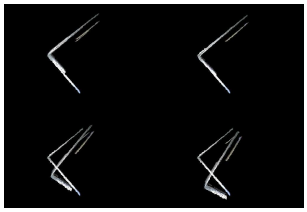
(g) pair $\mathcal{F}_{10}$ and $\mathcal{F}_{11}$



(h) pair $\mathcal{F}_{82}$ and $\mathcal{F}_{83}$



(i) pair $\mathcal{F}_{200}$ and $\mathcal{F}_{201}$



(j) pair $\mathcal{F}_{143}$ and $\mathcal{F}_{144}$

Figure 6.2: Registration using presented algorithms over selected 10 pairwise frames. The lower left: registration results using odometry; the lower right : registration results by ICP: upper left: registration results by SACIA; upper right: plane-matching registration

pair	ICP	SAC-IA	Plane Matching
$(\mathcal{F}_1, \mathcal{F}_2)$	$\times$	$\times$	$\surd$
$(\mathcal{F}_{21}, \mathcal{F}_{22})$	$\surd$	$\surd$	$\surd$
$(\mathcal{F}_{105}, \mathcal{F}_{106})$	$\times$	$\times$	$\surd$
$(\mathcal{F}_{141}, \mathcal{F}_{142})$	$\times$	$\surd$	$\surd$
$(\mathcal{F}_{249}, \mathcal{F}_{250})$	$\surd$	$\times$	$\surd$
$(\mathcal{F}_0, \mathcal{F}_1)$	$\surd$	$\surd$	$\surd$
$(\mathcal{F}_{10}, \mathcal{F}_{11})$	$\surd$	$\surd$	$\surd$
$(\mathcal{F}_{82}, \mathcal{F}_{83})$	$\surd$	$\surd$	$\surd$
$(\mathcal{F}_{143}, \mathcal{F}_{144})$	$\surd$	$\times$	$\surd$
$(\mathcal{F}_{200}, \mathcal{F}_{201})$	$\times$	$\surd$	$\surd$

Table 6.1: Comparison of pairwise registration using different algorithms, i.e., ICP, SAC-IA and plane matching registration

two parts by a door on a small hallway connecting two parts of the building. For reason of convenience, we call the two parts of the testing corridor left and right part of the corridor respectively. Figure 6.3 shows different parts of the real scenario. In order to see if the robot may recognize the same places visited previously, in this experiment, a looped trajectory is required. In our experiment the robot started from the left side of the corridor going along the corridor up to the right side, then it returned to the starting location, in order to form a loop closing. The robot traveled autonomously using a simple obstacle avoidance algorithm. During the run the robot captured 258 point clouds and covered an area of 35 m$\times$4 m.

(a) Left part of the corridor (b) Right part of the corridor

Figure 6.3: The real scenario of the corridor.

Figure 6.4 shows a comparison between the robot trajectory estimated by odom-

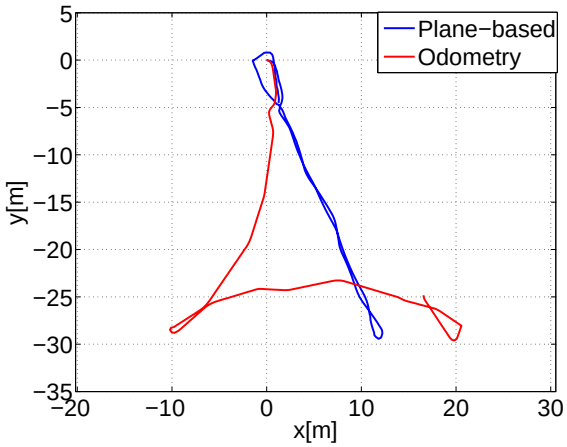


Figure 6.4: Comparison of estimated trajectories. The red line presents the recorded trajectory by the robot odometry, while the blue one shows the optimized trajectory (obtained with the proposed plane-based approach). The starting position is set to $(0, 0)$. Apparently, the drifting error accumulates resulting in a curved odometry trajectory. Meanwhile, according to odometry trajectory, the robot did not go back to its starting position, which is not true. Therefore, a big divergence happens. While the estimated trajectory using presented approach is much more consistent.

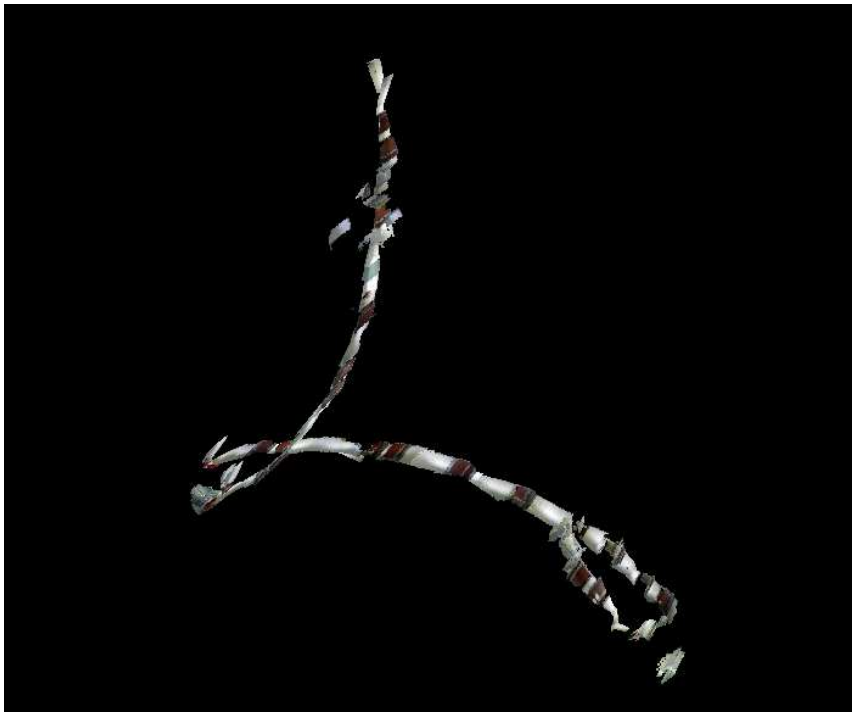


Figure 6.5: Top view of 3D map obtained using odometry only. Note that the obtained map presents a seriously curved corridor, corresponding to the odometry trajectory shown in Figure 6.4. Obviously, it does not match the real scenario.

Figure 6.6: Top view of the long corridor experiment reconstructed using the presented plane-based SLAM. Note that it is more consistent and its shape is much close to the real scenario.

etry only and by our plane-based SLAM algorithm. In this figure, the blue line shows the obtained robot trajectory using plane-based SLAM, against the odometry measured path which is plotted in red. Apparently, as shown from Figure 6.4, the trajectory provided by wheel odometry is not consistent, as the robot does not return to its starting position.

The 3D maps are constructed by attaching each acquired point cloud to its associated robot pose. Figure 6.5 shows the 3D map obtained by registering the point clouds just using the robot odometry. Corruption of the map is clearly observable through the deviation of corridor toward one side as the robot continues its explorations. Unbounded odometry drift is the reason of this deviation. Consequently, at the end the robot is totally lost by an error of approximately 35 m from the true position.

In contrast, in Figure 6.6 the map obtained by the plane-based SLAM algorithm seems consistent with regard to the real scenario. It can be seen that the shape and orientation of the constructed map is precisely matching the real scenario. The resulted 3D map clearly represents the main structures of the corridor. Walls and doors are correctly mapped compared to those in reality. This is of course theoretically expected, since using plane matching procedure, there is no error on orientation. Notice that at the right end of the constructed corridor, small pieces of the corridor are missed. That is because the robot just passes by the scenes only once in this experiment, and the detected planes with areas less than a pre-defined threshold or with few supporting points, are ignored.

6.2.2 Scenario 2: A Hall

To test the presented plane-based SLAM approach further, the second experiment was carried out in a closed small hall, in the third floor of the DAUIN building. The environment is similar to a typical office environment including walls, doors, and also glass windows, tables, chairs, boxes and other dynamic objects. Figure 6.7 shows the real appearance of the close hall.

The robot explored around the inner surrounding walls once such that it was able to capture the structure and shape of the environment completely. At the end, the robot returned to its starting point to form a loop, covering a total distance about 30 m. 189 3D point clouds were taken when the robot was moving continuously, one approximatively every 0.2 meters.

Similar to the previous experiment, the estimated trajectories are shown as well as the obtained 3D maps. Figure 6.8 presents the odometry trajectory and the optimized trajectory using plane-based SLAM algorithm, and their corresponding 3D maps are shown in Figure 6.9 and Figure 6.10, respectively.

As expected, the robot accumulates an error when it is moving, leading to an inconsistent map of the environment, shown in Figure 6.9. It is easy to observe that at first, the odometry is still good in this experiment since during this part, the robot mainly made a forward motion (the robot started from the upper right part of the hall). While after a big turn, the accumulated error is easily visible as in the reconstructed map.

Figure 6.10 shows the reconstruction of the environment using plane-based SLAM method. In comparison to Figure 6.9, the remaining accumulated error seems to be negligible. The reconstructed map seems consistent and its main structure precisely matches the real scenario. Note that some alignment errors are still present and some details are missing. This is mainly due to the fact that few corners are detected because of the lacking of big supporting planes, since the environment is more cluttered with respect to the one in the previous experiment. However, the fact that corresponding planes remain parallel shows that the rotation was accurately estimated.

6.2.3 Scenario 3: A Large Loop

In the next experiment, the algorithm is challenged against a large loop to evaluate the performance of presented plane-based SLAM further. The experiment is done in the ground floor of the main building of Politecnico di Torino. The experimental zone approximately consists in three main corridors in two directions. Classrooms and laboratories are along the both sides of the corridors. For the convenience of

(a) Left part of the hall (b) Right part of the hall

Figure 6.7: The real appearance of the hall.

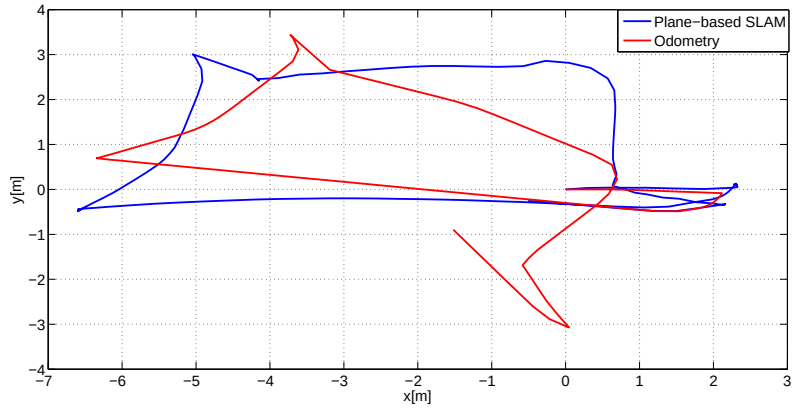


Figure 6.8: The estimated trajectories for the second experiment. As before, the starting position of robot is set to be $(0, 0)$. The red one presents the odoemtry path which is not globally consistent. While the blue path shows the obtained trajectory using plane-based SLAM.

Figure 6.9: A 3D view of the obtained map using odometry only. The drifting error and inconsistency are easily observable. Specially the walls at the right part of the resulting map are same regions, while they are presented in different locations.

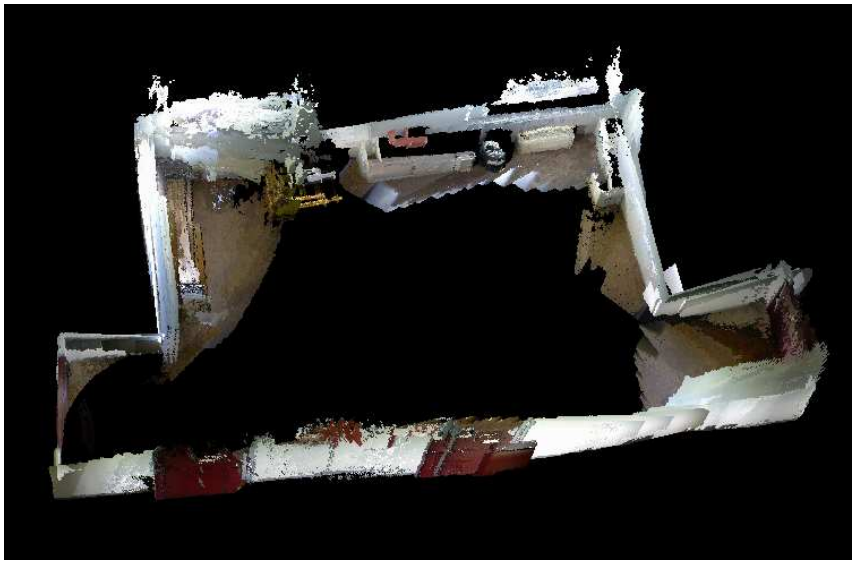


Figure 6.10: A 3D view of the built map using plane-based SLAM. Apparently, it closely presents the main structures of the real scenario, even though some small misalignment are still present.

explanation, the corridors are labeled as A, B, and C respectively, and their configuration is shown in Figure 6.11. The corridors are perpendicular or parallel to each other. It is important to note that Figure 6.11 only approximately presents the relative relationship between the corridors, but not the ground truth of the environment. The real appearance of these three corridors are shown in Figure 6.13, respectively.

During the experiment, most of the time the robot explored automatically. While in order to scan the environment totally or to form local loops, when the robot was arriving at the intersection area of two corridors, the robot was manually guided to enter into the expected corridor. Meanwhile, in order to make sure that successfully paired features can be detected, the robot was manually moved through at a low speed, since when the robot moves into a new room, the number of newly observed features will be relative high, whereas most of the old features go out of the view.

The robot started moving along the corridor A, and turned left to travel into corridor B. After exploring in corridor B, the robot returned to corridor A and formed the first local loop. To capture the structure of corridor C, the robot explored corridor A and B again, then entered into the corridor C. After exploring it some time, the robot returned back to corridor A. At the end, the robot stopped at the middle part of the corridor B. A manual path of the robot is described in Figure 6.12.

The robot traveled a total distance of roughly 180 m and collected 651 sensor samples, covering an area about 15m$\times$20m.

As described before, during this test, two local loops, 1-2-3-4-5-1, 1-2-3-6-7-8-4-5-1 presented in red line and magenta line in 6.12 respectively, and one global loop 1-2-3-4-5-1-2-3-6-7-8-4-5-1-2 are formed. Actually the local loop closings are helpful to reduce the accumulated error, especially when the robot traveled a long distance. That is theoretically expected, since when the robot returns back into a known environment, it relocalizes itself, allowing to reduce the accumulated error.

The estimated trajectories using odometry trajectory only and plane-based SLAM, is presented in Figure 6.14. And their corresponding constructed maps are shown in Figure 6.16 and Figure 6.17, respectively. Since in this experiment, the robot traveled longer than the first two experiments, the odometry drift is much larger at the end as expected. Actually before the robot performs the first local loop, the robot was totally lost as it can be seen from the red plotted path, shown in Figure 6.14. Therefore, its corresponding map is quite messed up. As apparent from the Figure 6.16, the odometry map is totally corrupted and the main structures of the corridors can not be observed.

To show the effectiveness of the proposed loop closure detection using VFH, discussed in Chapter 5, the robot trajectory is estimated using plane-based SLAM but without loop closings, shown in Figure 6.15. In Figure 6.15, each blue dash

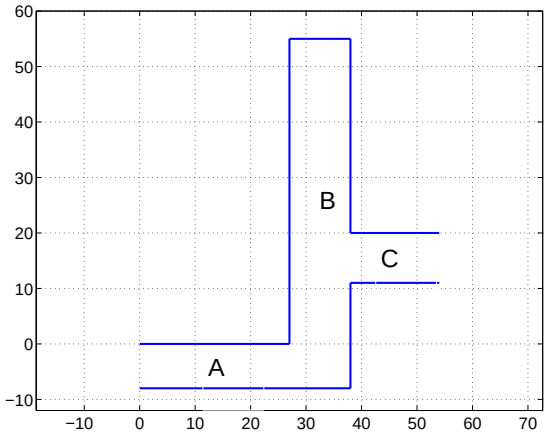


Figure 6.11: The configuration of the testing corridors. The corridors are perpendicular or parallel to each other, and labeled as A, B and C respectively.

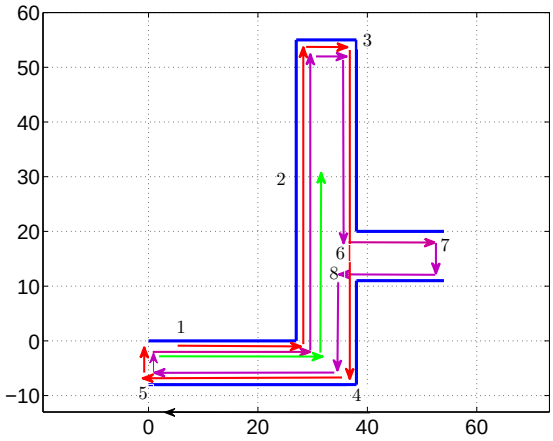


Figure 6.12: The manual path of the robot. The robot starts moving from position 1. After exploring the three corridors, it stops at position 2. During its exploration, besides a global loop, two local loops are formed, presented in red and magenta lines respectively.

(a) Corridor A (b) Corridor B (c) Corridor C

Figure 6.13: The real appearance of corridors in the large loop environment.

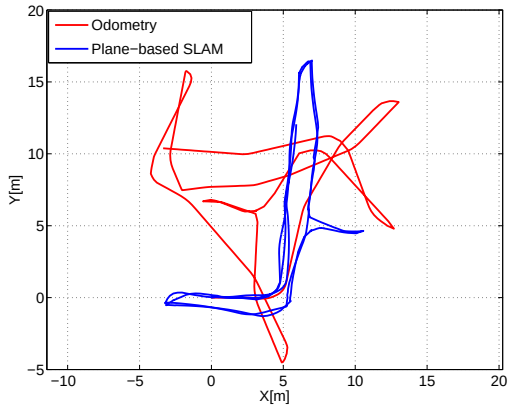


Figure 6.14: The obtained trajectories for the large loop experiment. As depicted in above two experiments, the red path are obtained by the wheel odometry. Obviously, the robot is totally lost and the trajectory is quite messed up. While the blue one shows the optimized trajectory using plane-based SLAM. As it can be seen, it seems consistent. Its overall shape is similar to the configuration of the experimental three corridors, shown in Figure 6.11.

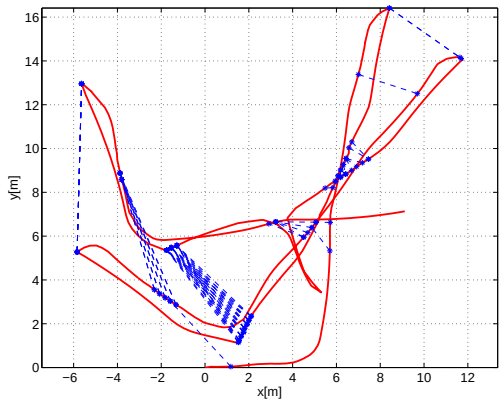


Figure 6.15: The estimated trajectory using plane-based SLAM approach but without loop closings. The detected loop closures in this experiment are shown in blue dash lines. Each blue line connected the detected loop closure frames together. The loop closures are obtained using the loop closure detection algorithm based on VFH discussed in Chapter 5.

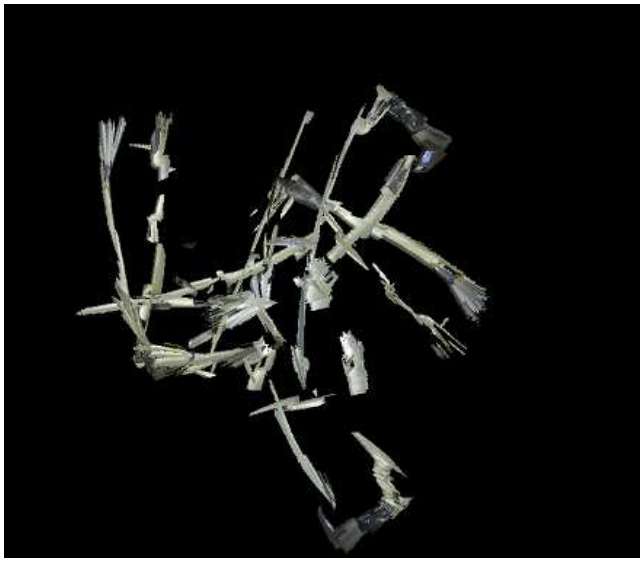


Figure 6.16: The 3D map obtained using odometry only. Obviously, it is a mess. The structures of the experimental corridors are totally corrupted.

Figure 6.17: The 3D map constructed using plane-based SLAM. Its shape and orientation is much close to the real scenario. It clearly presents the main structures, e.g., walls, doors, pillars etc.

line presents a detected loop closure and it connects the related two non-successive poses together. By comparing the estimated trajectory without loop closings, plotted in red in Figure 6.15, with the optimized trajectory with loop closings, the blue path in Figure 6.14, it can be observed that the remained drifted error in Figure 6.15, is almost eliminated in Figure 6.14, as seen from its corresponding 3D map, shown in Figure 6.17.

As it can be seen in Figure 6.17, the obtained 3D map closely matches the real experimental scenarios, especially from the orientation point of view. Note that the detected walls, doors and pillars are correctly mapped through the environment. In general the estimated features are well overlapped. This is of course theoretically expected, since using the plane matching registration, which correct the orientation error by finding the corresponding features in consecutive frames. While for the frames concluding the same corner features, the relative poses error in both orientation and position can be corrected fully. Meanwhile the use of loop closures allows the accumulated error is diminished.

However, it is also evident in the resulting map in Figure 6.17 that at the left side of corridor A, some orientation error remains and the walls on two sides are not parallel. The main reason for this is that the corridor A is not a closed corridor. When the robot turned, no features could not be detected. Therefore the frames cannot be matched to the prior ones. In this work, if a frame can not be matched, it will be connected to the prior node in the pose graph using the odometry information. However, as we all known, when the robot turns a big angle, the error of relative pose changes is high. Therefore, the orientation error between these frames could not be corrected totally.

We also notice that there are some misalignments between scans of some surfaces that were detected in multiple frames, indicating a relative error between the poses in the graph from which the point clouds were taken. For instance, at the beginning part of corridor C, the surfaces are not overlapped very well. That is mainly due to the fact that few corners are detected during because of the noise and occlusion in the point clouds. Another reason for this is the dominance of window glass on one wall of the corridor C. Such big windows are mostly invisible to the Kinect sensor, and even worse than that, it brings points with high noise, from the structures inside the office, which is along the corridor C.

6.3 Summary

This Chapter presented the experimental validation of the 3D mapping algorithm presented above.

At first, to test the robustness and correctness of the presented plane matching algorithm, experiments for registering pairwise scans were presented. By comparing it to the standard ICP and SACIA global alignment, the presented algorithm is shown to be more robust and accurate.

Then to evaluate the performance of the presented plane-based SLAM algorithm, three experiments were carried out in the third floor of DAUIN and the main building of Politecnico Di Torino. The reconstructed 3D maps of the experimental environments are consistent and close to the real scenarios, especially from the orientation point of view, which prove the presented algorithm is able to construct the explored environments in a consistent manner.

Chapter 7

Conclusions and Outlook

7.1 Conclusions

This short Chapter provides a brief outlook on the issues discussed in the thesis and highlights major conclusions and future directions of research regarding the major concerns of this work.

The goal of this thesis was to build the 3D maps for structured 3D indoor environments by developing a complete plane-based SLAM approach and validate it experimentally. This firstly required to find 3D sensors suitable for a mobile robot. Typically, laser range finders and depth cameras were used for 3D SLAM approaches in order to acquire dense point clouds. Recently, Microsoft Kinect has dominated the stage of 3D robotic sensing, as a low-cost, low-power sensor, that is able to acquire color and depth images at high frame rate. In our case, instead of expensive laser sensors, a Microsoft Kinect was employed as the exteroceptive sensor. For each frame, it is able to delivering dense 3D data composed of 307,200 points. Robot odometry was also used to initialize the search for correspondences between observations between consecutive frames.

Map building requires a known pose. It can be decomposed into three basic pieces, each of them are critical to building a globally consistent map successfully. The first is estimating the spatial constraint between consecutive frames, which is addressed as scan-matching problem. The second challenge is loop closing: recognizing when a robot has revisited a place it has been previously. Each loop closure represents a constraint on the trajectory of the robot. During the above two steps, the estimated spatial constraints are encoded in the edges between different nodes in a pose graph, whose nodes represent robot poses. Once such a pose graph is constructed, one seeks to find a configuration of the nodes that is maximally consistent with the measurements. This involves solving a large error minimization problem

and is referred as the third step of 3D mapping. The first two steps consists of pose graph construction, called SLAM front-end. While the third step is referred to be SLAM back-end.

It was discussed that abstracting raw point clouds into geometrical features leads to more efficient SLAM while at the same time more compact and structurally informative representations are obtained which greatly enhance interaction of the robot with its environment. Therefore, feature based SLAM was selected. In this thesis, the mobile robot was working in an indoor environment. Based on the fact that in indoor environments, several structures like doors, walls, tables, ground floor, etc., can be modeled as planar surface patches, which are parallel or perpendicular to each other. Therefore, planar patches have been found to be a good feature for 3D visual SLAM, while also being a quite good representation for the final 3D map. In addition, orthogonal corners, constructed by three intersecting perpendicular planes, are more distinguishable and considered higher features.

To extract the planar surfaces and 3D orthogonal corners from the raw sensor data robustly and accurately. The popular RANSAC plane model was iteratively executed to find planar surfaces in the scene, returning the plane with the most inliers from the 3D point cloud. In order to ensure that the obtained planar surfaces present the real geometry of the environment precisely, a distance-based clustering procedure and merging procedure were applied to refine the plane extraction results. This is of vital importance in mapping, since the followed data association is directly dependent on the accuracy of the extracted features. Based on the extracted planes, a 3D corner was formed by three intersecting planes, which are perpendicular to each other. The experiment results shows that the planes and 3D corners in the sensor data can be detected effectively.

The main original contribution of this work consists of finding correspondences between planar surfaces, as well as 3D orthogonal corners in the consecutive frames. After the correspondences have been decided on, the relative rotation and translation that aligns the corresponding set of features are computed. The estimated roto-translation encodes the pose changes of the robot between the related frames, which is then added to a pose graph. To robustly determine corresponding planes in different frames robustly is difficult, since wrong matches will result in a big divergence in the trajectory of robot. In Chapter 4, a plane matching algorithm was presented for determining the unknown plane correspondences by maximizing the similarity metric between a pair of planes within a search space. The search space is pruned using the followed criteria: odometric rotation agreement test, odometric translation agreement test, texture similarity test, and size similarity test. To discard more wrong matches, a plane matching consistent test is given to determine the resolved

correspondences. Then the determined plane correspondences were extended to the corner matching procedure.

Loop closing is a form of place recognition that is central to the task of map building. A successful loop closing allows to eliminate the accumulated drifting error when the robot moves, and prevents re-mapping of the same location in a wrong metric location. In this work, a loop closure detection algorithm from 3D point clouds by comparing their VFH descriptors and color histograms. Compared to 3D point clouds, VFH descriptors compress the input point clouds geometry information into meaningful statistic histograms while keeping the viewpoint component, thus it reduces the dimension and store space of the data. To reduce the computation time, we defined the frame in which corners are included as keyframes. Each time a keyframe is found, we compare its VFH feature and color histogram with the previous keyframes. A closure is detected if enough geometrically and color appearance consistent between pairwise frames matching. Then the relative roto-translation between the related pairwise frames were estimated by the corresponding corners in these frames, where the corner correspondences are determined using VFH feature again. As it can seen, this approach uses only the appearance of 3D point clouds to detect loops and requires no pose information.

The pairwise transformations between sensor poses, form the edges of a pose graph, which is referred to as SLAM front-end. However, due to the estimation errors, the edges form no globally consistent trajectory. To create a globally consistent trajectory, a linear approximation was used to optimize the obtained pose graph. Then the 3D map of the environment is constructed by attaching each acquired point cloud to the corresponding pose estimate.

The plane-based SLAM approach was evaluated in three different scenarios. Among them, two experiments were carried out in the third floor of DAUIN, Politecnico di Torino, and the third one was performed in the main building of Politecnico di Torino. The experiments were carried out with a differential-drive mobile robot of our lab equipped with a Microsoft Kinect and a SICK laser sensor, as described before. The experiment results showed that the system was able to reconstruct all test environments in an consistent way. The reconstructed 3D maps were close to the real scenarios and were able to present the structures of the test environment precisely. Meanwhile, we evaluated the plane matching pairwise registration, and compared it with the standard ICP and SACIA with FPFH. The experiment results showed that the overall performance of plane matching registration was better than the ICP and SACIA.

7.2 Future Work

Despite the encouraging experiment results presented in this thesis, there are still different aspects that could be improved. In the future, the following issues are going to be considered.

1. The first issue concerns to improve the discernibility of the planar features used with the goal to simplify plane matching algorithm at the same time improve the robustness of the plane matching.

2. During the experiments, we noticed that when the robot enters into a new area, or the robot rotates a big angle, more new features come into the view, while few old features remain. Therefore, it is difficult to find corresponding features such that the relative pose change error could not be corrected for these related frames. An alternative solution for this is to use two Microsoft Kinect cameras to capture the environment of different views.

3. It is of foremost interest for plane-based mapping to concentrated on the extraction of a few, large planes per frame that give high confidence correspondences so to estimate the relative orientation and position information for related frames. To make use the reconstructed 3D map, especially in path planning and navigation, it is better to represent the 3D maps with surface patches, especially compared to point clouds, since large planar patches with polygon boundaries are very well suited for computational geometry algorithms employed in path planning and navigation.

4. The algorithm presented in this thesis is limited in the structured indoor environment. It is hence of interest to consider higher order surfaces, e.g., quadrics, as representations for large surface patches in other environment.

7.3 Publication

- L. Carlone, J. Yin, S. Rosa, and Z. Yuan, Graph optimization with unstructured covariance: fast, accurate, linear approximation, in Proc. of the Int. Conf. on Simulation, Modeling, and Programming for Autonomous Robots, 2012.

Bibliography

[1] J. Bauer, K. Karner, A. Klaus, and R. Perko. Robust range image registration using a common plane. In *Proceedings of 2004 WSCG*, 2004.

[2] H. Bay and A. Ess. SURF: Speeded up robust features. *Computer Vision and Image Understanding (CVIU)*, 110(3):346–359, 2008.

[3] J. Berkmann and T. Caelli. Computation of surface geometry and segmentation using covariance techniques. *IEEE Transactions on Pattern Analysis Machine Intelligence*, 16(11):1114–1116, 1994.

[4] P. J. Besl and N. D. Mckay. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992.

[5] P. Biber, H. Andreasson, T. Duckett, and A. Schilling. 3d modeling of indoor environments by a mobile robot with a laser scanner and panoramic camera. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sendai, Japan, 2004.

[6] P. Biber and W. Strasser. The normal distribution transform: a new approach to laser scan matching. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2743C–2748, Las Vegas, USA, 2003.

[7] J. Borenstein and L. Feng. Measurement and correction of systematic odometry errors in mobile Robots. *IEEE Transactions on Robotics and Automation*, 12(5):869–880, 1996.

[8] M. C. Bosse and R. Zlot. Map matching and data association for large-scale two-dimensional laser scan-based SLAM. *International Journal of Robotics Research*, 27(6):667–691, 2008.

[9] G. Bradski. The OpenCV Library. *Dr.Dobb's Journal of Software Tools*, 2000.

[10] J. Callmer, K. Granström, J. I. Nieto, and F. T. Ramos. Tree of words for visual loop closure detection in urban SLAM. In *Proceedings of the Australian Conference on Robotics and Automation (ACRA)*, Canberra, Australia, December 2008.

[11] L. Carlone, R. Aragues, J. A. Castellanos, and B. Bona. A first-order solution to simultaneous localization and mapping with graphical models. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1764–1771, 2011.

[12] L. Carlone, R. Aragues, J. A. Castellanos, and B. Bona. A linear approximation for graph-based simultaneous localization and mapping. In *Proc. of Robotics: Science and Systems*, 2011.

[13] L. Carlone, J. Yin, S. Rosa, and Z. Yuan. Graph optimization with unstructured covariance: fast, accurate, linear approximation. In *Proceeding of the International Conference on Simulation, Modeling, and Programming for Autonomous Robots*, 2012.

[14] J. A. Castellanos, J. M. M. Montiel, J. Neira, and J. D. Tardos. The spmap: A probabilistic framework for simultaneous localization and map building. *IEEE Transactions on Robotics and Automation*, 15, 1999.

[15] Y. Chen and G. Medioni. Object modelling by registration of multiple range images. *Image Vision Computing*, 10(3):144–155, 1992.

[16] M. V. C. S. . Z. S. CSEM, The SwissRanger. `http://www.swissranger.ch`, 2006.

[17] M. Cummins and P. Newman. Fab-map: Probabilistic localization and mapping in the space of appearance. *The International Journal of Robotics Research*, 27(6):647–665, 2008.

[18] M. Cummins and P. Newman. Highly scalable appearance-only SLAM-FAB-MAP 2.0. In *Proceedings of Robotics: Science and Systems (RSS)*, Seattle, USA, June 2009.

[19] F. Dellaert and M. Kaess. Square root SAM: simultaneous localization and mapping via square root information smoothing. *International Journal of Robotics Research*, 25(12):1181–1203, 2006.

[20] T. Duckett, S. Marsland, and J. Shapiro. Fast, on-line learning of globally consistent maps. *Autonomous Robots*, 12(3):287–300, 2002.

[21] E. Eade and T. Drummond. Unified loop closing and recovery for real time monocular slam. In *Proceedings of the British Machine Vision Conference*, Leeds, United Kingdom, September 2008.

[22] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard. An evaluation of the RGB-D SLAM system. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2012.

[23] R. Eustice, H. Singh, and J. Leonard. Exactly sparse delayed-state filters. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2428–2435, Barcelona, Spain, 2005.

[24] D. Fischer and P. Kohlhepp. 3d geometry reconstruction from multiple segmented surface descriptions using neuro-fuzzy similarity measures. *Journal of Intelligent and Robotic Systems*, 29(4):389–431, 2000.

[25] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[26] J. Folkesson and H. Christensen. Graphical SLAM - a self-correcting map. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, New Orleans, LA, USA, 2004.

[27] U. Frese, P. Larsson, and T. Duckett. A multilevel relaxation algorithm for simultaneous localisation and mapping. *IEEE Transactions on Robotics*, 21(2):196–207, 2005.

[28] U. Frese and L. Schröder. Closing a million-landmarks loop. In *Proceedings of IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 5032–5039, 2006.

[29] N. Gelfand, L. Ikemoto, S. Rusinkiewicz, and M. Levoy. Geometrically stable sampling for the ICP algorithm. In *Fourth International Conference on 3D Digital Imaging and Modeling (3DIM)*, pages 260–270, 2003.

[30] G. Godin, D. Laurendeau, and R. Bergevin. A method for the registration of attributed range images. In *Proceedings of Third International Conference on 3D Digital Imaging and Modeling*, 179–186, 2001.

[31] R. C. Gonzalez and R. E. Woods. *Digital image processing*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2rd edition, 2001.

[32] K. Granström, J. Callmer, F. Ramos, and J. Nieto. Learning to detect loop closure from range data. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 15–22, 2009.

[33] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard. A tutorial on Graph-based SLAM. *IEEE Transactions on Intelligent Transportation Systems Magazine*, 2(4):31–43, 2010.

[34] G. Grisetti, R. Kummerle, C. Stachniss, and C. Hertzberg. Hierarchical optimization on manifolds for online 2d and 3d mapping. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 273–278, 2010.

[35] G. Grisetti, C. Stachniss, and W. Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Transactions on Robotics*, 23(1):34–46, 2007.

[36] G. Grisetti, C. Stachniss, and W. Burgard. Non-linear constraint network optimization for efficient map learning. *IEEE Transactions on Intelligent Transportation Systems*, 10(3):428–439, 2009.

[37] G. Grisetti, C. Stachniss, and W. Burgard. Non-linear constraint network optimization for efficient map learning. *IEEE Transactions on Intelligent Transportation Systems*, 10(3):428–439, 2009.

[38] D. H”ahnel, W. Burgard, D. Fox, and S. Thrun. An efficient FastSLAM algorithm for generating maps of large-scale cyclic environments from raw laser range measurements. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 206–211, Las Vegas, NV, USA, 2003.

[39] D. Hähnel, W. Burgard, and S. Thrun. Learning compact 3d models of indoor and outdoor environments with a mobile robot. *Robotics and Autonomous Systems*, 44(1):15–27, 2003.

[40] A. Harati. *Simultaneous Localization and Mapping for Structured Indoor Environments*. PhD thesis, Eidgenössische Technische Hochschule Zürich, 2008.

[41] A. Harati and R. Siegwart. Orthogonal 3D-SLAM for indoor environments using right angle corners. In *Proceedings of the 3rd European Conf. Mobile Robotics (ECMR'07)*, 2007.

[42] P. Henry, M. Krainin1, E. Herbst, X. Ren, and D. Fox. RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments. In *Proceedings of the International Symposium on Experimental Robotics (ISER)*, 2010.

[43] C. Hertzberg, R. Wagner, U. Frese, and L. Schröder. Integrating generic sensor fusion algorithms with sound state representations through encapsulation of manifolds. *Information Fusion*, 14(1):57–77, 2011.

[44] K. Ho and P. Newman. Combining visual and spatial appearance for loop closure detection in SLAM. In *Proceedings of European Conference on Mobile Robots*, Ancona, Italy, September 2005.

[45] J. Horn and G. Schmidt. Continuous localization of a mobile robot based on 3D-laser-range-data, predicted sensor images, and dead-reckoning. *Robotics and Autonomous Systems*, 14(2–3):99–118, 1995.

[46] A. Howard, M. Mataric, and G. Sukhatme. Relaxation on a mesh: a formalism for generalized localization. pages 1055–1060, 2001.

[47] J. Jaw and T. Chuang. Registration of LIDAR point clouds by means of 3D line features. *Journal of the Chinese Institute of Engineers*, 2007.

[48] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert. iSAM2: incremental smoothing and mapping with relinearization and incremental variable reordering. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[49] M. Kaess, A. Ranganathan, and F. Dellaert. iSAM: incremental smoothing and mapping. *IEEE Transactions on Robotics*, 24(6):1365–1378, 2008.

[50] K. Klasing, D. Althoff, D. Wollherr, and M. Buss. Comparison of surface normal estimation methods for range sensing application. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Kobe, Japan, 2009.

[51] P. Kohlhepp, G. Bretthauer, M. Walther, and R. Dillmann. Using orthogonal surface directions for autonomous 3d-exploration of indoor environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3086–3092, 2006.

[52] P. Kohlhepp, P. Pozzo, M. Walther, and R. Dillmann. Sequential 3D-SLAM for mobile action planning. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2004.

[53] K. Konolige. Large-scale map-making. In *Proceedings of the 19th national conference on Artifical intelligence (AAAI)*, pages 457–463, San Jose, CA, 2004.

[54] K. Konolige and M. Agrawal. FrameSLAM: from bundle adjustment to real-time visual mapping. *IEEE Transactions on Robotics*, 24(5):1066–1077, October 2008.

[55] K. Konolige, J. Bowman, J. D. Chen, P. Mihelich, M. Calonder, V. Lepetit, and P. Fua. View-based maps. *International Journal of Robotics Research (IJRR)*, 29(8):941–957, 2010.

[56] K. Konolige, J. Bowman, J. D. Chen, P. Mihelich, M. Calonder, V. Lepetit, and P. Fua. View-based maps. *The International Journal of Robotics Research*, 29(8):941–957, 2010.

[57] K. Konolige, G. Grisetti, R. Kümmerle, W. Burgard, B. Limketkai, and R. Vincent. Efficient sparse pose adjustment for 2D mapping. In *Proceedings of the IEEE/RSJ Internation Conference on Intelligent Robots and Systems (IROS)*, 2010.

[58] J. Krumm. Intersection of two planes. Technical report, Microsoft Research, Redmond, WA, USA.

[59] R. Kümmerle, G. Grisetti, and W. Burgard. Simultaneous calibration, localization, and mapping. In *Proceedings of the IEEE/RSJ Internation Conference on Intelligent Robots and Systems (IROS)*, pages 3716–3721, 2011.

[60] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g^2o: A general framework for graph optimization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, 2011.

[61] P. Lamon, A. Tapus, E. Glauser, and N. Tomatis. Environmental modeling with fingerprint sequences for topological global localization. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 3781–3786, 2003.

[62] J. Leonard and H. Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *IEEE Transactions on Robotics and Automation*, 7(3):376–382, 1991.

[63] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[64] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robotics*, 4:333–349, 1997.

[65] M. Magnusson, H. Andreasson, A. Nüchter, and A. J. Lilienthal. Automatic appearance-based loop detection from 3D laser data using the normal distributions transform. *Journal of Field Robotics*, 26(11–12):892–914, 2009.

[66] P. C. Mahalanobis. On the generalised distance in statistics. In *Proceedings National Institute of Science, India*, volume 2, pages 49–55, 1936.

[67] Microsoft. `http://www.xbox.com/en-US/kinect`, 2010.

[68] M. J. Milford and G. F. Wyeth. Mapping a suburb with a single camera using a biologically inspired SLAM system. *IEEE Transactions on Robotics*, 24(5):1038–1053, 2008.

[69] A. Murillo, J. Guerrero, and C. Sagues. SURF features for efficient robot localization with omnidirectional images. In *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 3901–3907, 2007.

[70] E. Olson, J. Leonard, and S. Teller. Fast iterative optimization of pose graphs with poor initial estimates. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2262–2269, 2006.

[71] K. Pathak, A. Birk, N. Vaskevicius, M. Pfingsthorn, S. Schwertfeger, and J. Poppinga. Online three-dimensional slam by registration of large planar surface segments and closed-form pose-graph relaxation. *Journal of Field Robotics*, 27(1):52–84, 2010.

[72] K. Pathak, A. Birk, N. Vaskevicius, and J. Poppinga. Fast registration based on noisy planes with unknown correspondences for 3D mapping. *IEEE Transactions on Robotics*, 26(3):424–441, 2010.

[73] R. Paul and P. Newman. FAB-MAP 3D: topological mapping with spatial and visual appearance. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2649–2656, 2010.

[74] P. T. PhotonIC (R) PMD 3k S. `http://www.pmdtec.com`, 2010.

[75] F. Ramos, J. Nieto, and H. Durrant-Whyte. Recognising and modelling landmarks to close loops in outdoor SLAM. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2036–2041, 2007.

[76] R. B. Rusu, N. Blodow, and M. Beetz. Fast point feature histograms (FPFH) for 3D registration. In *Proceedings of the 2009 IEEE International Conference on Robotics and Automation(ICRA)*, 2009.

[77] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz. Aligning point cloud views using persistent feature histograms. In *Proceedings of the 21st IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2008.

[78] R. B. Rusu, G. Bradski, R. Thibaux, and J. Hsu. Fast 3D recognition and pose using the viewpoint feature histogram. In *Proceeding of the International Conference on Intelligent Robots and Systems (IROS)*, 2010.

[79] R. B. Rusu and S. Cousins. 3D is here: Point Cloud Library (PCL). In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, 2011.

[80] R. B. Rusu, Z. C. Marton, N. Blodow, and M. Beetz. Learning informative point classes for the acquisition of object model maps. In *Proceedings of the 10th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, 2008.

[81] Y. B. Shalom and X. Li. *Estimation and tracking: principles, techniques, and software*. Artech House, Boston, 1993.

[82] G. Sharp, S. Lee, and D. Wehe. ICP registration using invariant features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(1):90–102, 2002.

[83] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial relationships in robotics. In *Autonomous robot vehicles*. Springer-Verlag New York, 1990.

[84] I. Stamos and M. Leordeanu. Automated feature-based range registration of urban scenes of large scale. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 555–561, 2003.

[85] B. Steder, G. Grisetti, and W. Burgard. Robust place recognition for 3D range data based on point features. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1400–1405, Alaska, USA, 2010.

[86] B. Steder, R. B. Rusu, K. Konolige, and W. Burgard. Point feature extraction on 3D range scans taking into account object boundaries. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[87] H. Surmann, A. Nuechter, and J. Hertzberg. An autonomous mobile robot with a 3d laser range finder for 3D exploration and digitalization of indoor environments. *Robotics and Autonomous Systems*, 45(3–4):181–198, 2003.

[88] S. Thrun. Robotic mapping: A survey. In *Exploring Artificial Intelligence in the New Millenium*. Morgan Kaufmann, 2002.

[89] S. Thrun and M. Montemerlo. The GraphSLAM algorithm with applications to large-scale mapping of urban structures. *International Journal on Robotics Research*, 25(5/6):403–430, 2005.

[90] S. Thrun, Y.Liu, D. Koller, A. Ng, Z. Ghahramani, and H. F. Durrant-Whyte. Simultaneous localization and mapping with sparse extended information filters. *International Journal of Robotic Research*, pages 693–716, 2004.

[91] R. Wagner, O. Birbach, and U. Frese. Rapid development of manifold-based graph optimization systems for multi-sensor calibration and SLAM. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems(IROS)*, pages 3305–3312, 2011.

[92] J. Weingarten. *Feature-based 3D SLAM*. PhD thesis, EPFL,Lausanne, Switzerland, 2006.

[93] D. Wolter. *Spatial representation and reasoning for robot mapping-a shape-based approach*. PhD thesis, Bremen University, 2006.

[94] G. M. Y. Chen. Object modeling by registration of multiple range images. In *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on (1991)*, pages 2724–2729, 1991.